DOCUMENT RESUME

ED 154 774 IR 005 740

AUTHOR	Avner, Elaine
TITLE	Summary of TUTOR Commands and System Variables. Sixth Edition.
INSTITUTION	Illinois Univ., Urbana. Computer-Based Education Lab.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	Sep 77
CONTRACT	USNSF C-273
NOTE	130p.; For related document, see ED 124 130
EDRS PRICE	MF-$0.83 HC-$7.35 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; Higher Education; Instructional Systems; *Manuals; *Programing Languages
IDENTIFIERS	PLATO Computers; *TUTOR Programing

ABSTRACT

This summary is intended for the experienced author who needs a quick reference for the form of a tag and for the restrictions on the TUTOR language commands. Each command includes a brief description of its purpose and a description of the tag. The commands are grouped into the following categories: (1) calculating, (2) data keeping, (3) judging, (4) managing sites, (5) presenting, (6) routing, and (7) sequencing. The appendices include a list of the limits associated with commands and alphabetical lists of systems variables and commands. (DAG)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED154774

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

SUMMARY OF TUTOR® COMMANDS AND SYSTEM VARIABLES

(sixth edition)

Elaine Avner

PLATO Services Organization

Computer-based Education Research Laboratory

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

William James Strutz

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

IR005 740

5/3/15

Copyright© September 1977
by Board of Trustees
University of Illinois

First Edition May 1974
Second Edition June 1975
Third Edition November 1975
Fourth Edition August 1976
Fifth Edition May 1977
Sixth Edition September 1977

PLATO® and TUTOR® are service marks
of the
University of Illinois

All rights reserved. No part of this book may be reproduced in any form or by any means without permission in writing from the author.

This manuscript was prepared with partial support from the National Science Foundation (USNSF C-273) and the University of Illinois at Urbana-Champaign.

Acknowledgment

Many PLATO users have made suggestions on the form and content of this book: Allen Avner, Richard Blomme, Robert Bohn, George Friedman, James Ghesquiere, John Gilpin, William Golden, Bruce Sherwood, Judy Sherwood, Paul Tenczar, and Richard Videbeck.

Wayne Wilson and Roy Lipschutz assisted with the drafting work and final preparation of the manuscript.

This book was typed on a modified electric typewriter connected to a PLATO terminal. Steven Williams wrote the program which drives the typewriter.

This summary is intended for the exprienced author who needs a quick reference for the form of a tag and for some of the restrictions on commands. It is not designed to teach the fundamentals of TUTOR, nor to discuss TUTOR states, answer-judging, sequencing, etc. For such information authors may refer to "aids" and to The TUTOR Language by Bruce Sherwood.

Each command includes a brief description of its purpose and a description of the tag. The standard form is

command brief description

command DESCRIPTION OF TAG (any explanatory comments)

or

command actual tag

Note: Additional comments about this command.

NOTE: General comments about groups of commands.

The commands are grouped into seven categories: calculating (C), data keeping (D), judging (J), managing sites (M), presenting (P), routing (R), and sequencing (S). Each section is identified by the page number (e.g., C1 or P5) and by the mark at the edge of the page. By matching the marks at the edges of the index pages with the marks along the edges of the book, a user may quickly access the desired section.

Modifications to this book required by changes in the TUTOR language are contained in lesson "aids", option "Changes to 'Summary of TUTOR Commands'" (also reached via DATA: changes to summary). These modifications, along with other notes of interest, may be inserted in the spaces which have been provided between entries and on the additional pages at the end of each section.

CONTENTS

	Page
Abbreviations	1
Classification of commands and system variables	ii to viii
CALCULATING	
Basic calculating	C1
System operations and functions	C4
Random numbers	C7
Information	C9
Bit and character manipulation	C10
Operations on lists	C12
Data manipulation	C16
System variables for calculating	C24
DATA KEEPING	
Requesting data	D1
Classifying data	D2
Transferring data	D3
Student sign-on and sign-off	D6
System variables for data keeping	D7
JUDGING	
Preparation for responding	J1
Vocabulary lists	J4
Modification of judging copy of response	J5
Modification of judging procedure	J6
Storing judging copy of response	J8
Matching judging copy of response	J9
Information on specific words in response	J13
Unconditional judgment	J15
Reference to other units which may contain judging commands	J16
Alteration of judgment	J17
Alteration of feedback	J18
System variables for judging	J19
MANAGING SITES	
Site commands	M1
Station commands	M3
PRESENTING	
Basic display	P1
Graphics	P6
Relocatable graphics	P8
Drawing graphs	P10
Non-screen	P16
Special display	P18
System variables for presenting	P20
ROUTING	
Router lesson	R1
System router	R2
System variables for routing	R3
SEQUENCING	
Basic sequencing	S1
Automatic sequencing	S2
Key-initiated sequencing	S7
Timing	S9
Lesson connections and sections	S12
Lesson lists	S15
Lesson annotation and debugging	S17
System variables for sequencing	S20
Appendix	
Some limits associated with commands	A1
Keyset	A4
Keycodes, internal codes, alternate font memory locations	A5
Powers of two	A9
Alphabetical index	
System variables	I1
Commands	I2

C D J M P R S A I

Abbreviations and Notes

Below are listed the abbreviations used in the descriptions.

abbreviation	definition
arg	argument or tag entry
(b)	blank tag
coarse	character grid coordinates
CM	central memory
CPU	cental processing unit
disk	rotating magnetic disk storage unit
ECS	extended core storage
expr	mathematical expression
finex,finey	fine grid coordinates
num	number of
(opt)	optional argument
string	character string
var	variable
vars	variables

In conditional statements and in statements where a variable is set, suffixes m, Ø, 1, 2, etc., denote the minus condition, Ø condition, 1 condition, 2 condition, etc., e.g.,

```
match   VAR,WORDØ,WORD1,WORD2
do      EXPR,NAMEM,NAMEØ,NAME1,NAME2
```

In conditional statements the conditional expression is rounded (not truncated) to the nearest integer. Thus, a value of -.4 results in the Ø condition being selected rather than the minus condition.

Generally, wherever a tag entry may be a number, a mathematical expression will also be accepted.

Command names are enclosed in dashes when they are referred to in the descriptions, e.g., -next-. Key names are capitalized, e.g., NEXT.

Commands labeled "non-executable" are active only when the lesson is being condensed.

When variables are used in the tag of certain commands which require names in the tag, e.g., -area-, the variable must be enclosed in parentheses to indicate that the information needed is the contents of the variable and not a character string; e.g., -area (v3)- means the area whose name is contained in variable v3, while -area v3- means the area whose name is v3.

In the tags of some commands both commas and semicolons are used as separators, e.g., -draw-, -from-, -transfr-, conditional -jumpout-, -answerc-, etc. Correct placement of the separators is critical.

CALCULATING

Basic calculating C1

define
calc
calcc
calcs
addl
subl
zero
set

Random numbers C7

seed
randu
setperm
randp
remove
modperm

Information C9

clock
date
day
name
course
compute

Bit and character manipulation C10

search
pack
packc
itoa
move
otoa
htoa

Operations on lists C12

sort
sorta
finds
findsa
inserts
deletes
find
findall

Data manipulation C16

block
transfr
common
comload
comret
abort
commonx
initial
storage
stoload
dataset
datain
dataout
reserve
release
setname
getname
addname
rename
addrecs
delrecs
delname
names
backgnd
foregnd

System variables for calculating C24

lcommon
lstorag
zbpc
zbpw
zcpw
zcusers
zdsname
zdusers
zinfo
znscpn
znsmaxn
znxmaxr
znsnams
znsrecs
zrecs
zwpr

DATA KEEPING

Requesting data D1

dataon
dataoff

Classifying data D2

area
output
output1
setdat

Transferring data D3

readset
readd
readr

Student sign-on and sign-off D6

restart
finish

System variables for datakeeping D7

collecting data

dataon

session data

zsesset
zsesspt
zsessda

area data

aarea
aarrows
ahelp
ahelpn
aok
aokist
asno
aterm
atermn
atime
auno

JUDGING

Preparation for responding J1

eraseu
force
edit
arrow
arrowa
arheada
long
jkey
copy
endarrow

Vocabulary lists J4

list
endings
vocabs
vocab

Modification of judging copy of response J5

bump
put
putd
putv
close
loada

Modification of judging procedure J6

specs

Storing judging copy of response J8

store
storeu
storen
storea
open

Matching judging copy of response J9

match
answer
wrong
answerc
wrongc
concept
miscon
exact
exactc
exactv
ansv
wrongv
ansu
wrongu
touch
touchw
or
ans

Information on specific words in response J13

getword
getmark
getloc
compare

Unconditional judgment J15

ok
no
ignore

Reference to other units which may contain judging commands J16

join
iarrow
iarrowa

Alteration of judgment J17

judge

Alteration of feedback J18

okword
noword
markup

System variables for judging J19

judging in general

anscnt
ansok
jcount
judged
key
ntries
ztouchx
ztouchy

verbal responses

capital
entire
extra
order
phrase
spell
wcount

numerical responses

formok
opcnt
varcnt

MANAGING SITES

Site commands M1	Station commands M3
site set	station info
site info	station status
site active	station send
site stations	station logout
	station stopl
	station off
	station on

PRESENTING

Basic display P1

at
atnm
write
writec
show
showz
showt
showe
showo
showa
hidden
text
erase
mode
color
size
rotate
inhibit
delay
char
plot

Graphics P6

dot
draw
box
vector
window
circle
circleb

Relocatable graphics P8

rorigin
rat
ratnm
rdot
rdraw
rbox
rvector
rcircle

Drawing graphs P10

gorigin
axes
bounds
scalex
scaley
lscalex
lscaley
labelx
labely
markx
marky
polar
gat
gatnm
gdot
graph
gdraw
gbox
gcircle
gvector
vbar
hbar
delta
funct

Non-screen P16

slide
audio
play
record
enable
disable
ext
extout
saylang
say
sayc

Special display P.3

tabset
micro
charset
chartst
lineset
altfont

System variables
for presenting P20

mode
size
sizex
sizey
where
wherex
wherey

ROUTING

Router lesson R1

route
routvar
allow

System router R2

lesson
score
status

System variables for routing R3

errtype
ldone
lscore
lstatus
rcallow
router
rstart1
rstartu
rvallow

SEQUENCING

Basic sequencing S1

unit
entry

Automatic sequencing S2

jump
goto
do
join
exit
iferror
imain
branch
doto
if
elseif
else
endif
loop
endloop
outloop
reloop

Key-initiated sequencing S7

next, nextl
back, backl
stop
nextnow
nextop, nextlop
backop, backlop
help, helpl
data, datal
lab, labl
helpop, helplop
dataop, datalop
labop, lablop
term
termop
base
end

Timing S9

group
pause
collect
keytype
time
timer
timel
press
catchup
return
cpulim

Lesson connections and sections S12

use
jumpout
from
lessin
in
notes
cstart
cstop
cstop*

Lesson lists S15

leslist
addlist
removl
lname
findl

Lesson annotation and debugging S17

*
c
$$
change
step
*list

System variables for sequencing S20

args
backout
baseu
clock
fromnum
key
lessnum
lleslst
llesson
mainu
mallot
muse
nhelpop
proctim
ptime
sitenam
station
tactive
user
usersin
zaccnam
zcondok
zfroml
zfromu
zlesson
zpnfile
zpnotes
zretrnu
zreturn
zsnfile
zsnotes
zterm
ztouchx
ztouchy
zunit
zwpb

Basic calculating

define (non-executable) permits an author to rename variables and to define mathematical functions, arrays, and constants for a lesson and to specify those available for student use; defined variables must physically precede any reference to the variables in the lesson

for example:

```
define  SETNAME
        NAME1=v1,NAME2=n2,NAME3=65,NAME4=2(NAME1+NAME3),...
        FUNC(x,y,...)=some function of x, y, etc., where x, y, etc.
        are not already defined, although the expression on the right
        of the equal sign may contain previously defined names
           (up to 6 arguments are permitted)
```

The following definitions allow use of segmented variables.

```
segment,NAME=STARTING VAR,NUM BITS PER BYTE,s (opt)
segmentv,NAME=STARTING VAR,STARTING BIT POSITION,
                    NUM BITS PER BYTE,s (opt)
   (Starting variable address and byte size cannot be
   variables.  Byte size is from 1 to 59.  If the last
   argument is included, negative as well as positive integers
   may be stored.  In vertical segment, starting bit position
   may be from 1 to 60.)
```

The following definitions allow use of arrays.

```
array,NAME(SIZE)=STARTING VAR    (number of variables
                                  required equals size)
array,NAME(NUM ROWS,NUM COLUMNS)=STARTING VAR
   (number of variables equals rows x columns)
array,NAME(FIRST ELEMENT;LAST ELEMENT)=STARTING VAR
array,NAME(FIRST ROW ELEMENT,FIRST COLUMN ELEMENT;LAST
      ROW ELEMENT,LAST COLUMN ELEMENT)=STARTING VAR
```

Arrays may also be defined with segmented variables. The form is that for vertical segments. For example:

```
arraysegv,NAME(SIZE)=STARTING VAR,STARTING BIT POSITION,
          NUM BITS PER BYTE,s (opt)
```

Up to 255 elements are permitted in an array.

(-define- continued on next page.)

```
define  student
        all defines necessary for student responses, including units

        units,UNIT1,UNIT2,...   (maximum of 1Ø units, such as gram,
                                 meter, second,...)
           (The define set "student" may also include abbreviations
           and equivalences involving these units.  See -storeu-,
           -ansu-, and -wrongu- for these applications.)
```

To merge a previous set of definitions (SETA) with a set being defined at this point in the program (SETB):

```
define  SETB,SETA
        definitions in SETB
```

To purge previous define sets:

```
define  purge,SETNAME   (discards define set named)
define  purge   (discards all define sets except "student")
```

Note: Up to 4ØØ definitions are permitted in a define set, fewer if definitions are long. Defined names and names of define sets cannot exceed 7 characters, cannot contain mathematical operators, and must start with a letter.

calc assigns the value of the expression on the right side of the assign arrow to the variable or array on the left side, or packs up to 1Ø characters into an integer variable

for example:

```
calc    VAR← EXPR
calc    VAR← "STRING"   (right-justified, use n-variable)
calc    VAR← 'STRING'   (left-justified, use n-variable)
calc    ARRAYNAME← EXPR   (includes standard arithmetic operations,
           bit operations, logical operations, and array functions)
calc    VAR← ARRAYNAME1 ° ARRAYNAME2
calc    ARRAYNAME1← ARRAYNAME2 X ARRAYNAME3
```

Note: See section on SEQUENCING, Automatic sequencing for -doto-, -branch-, -if-, -loop-, and related directives. These are calc-type commands which allow branching within a unit.

calcc does one of several calculations depending on the rounded value of a conditional expression

```
calcc   EXPR,VAR1← EXPRM,VAR2← EXPRØ,VAR3← EXPR1,,VAR4← EXPR3
```

calcs sets a variable to one of several values depending on the rounded value of a conditional expression

```
calcs   EXPR,VAR← EXPRM,EXPRØ,EXPR1,EXPR2,,EXPR4
```

NOTE: Up to 61 calculations may be performed with -calcc- or -calcs-. A blank tag entry (,,) means no calculation is done for the corresponding value of the conditional expression.

addl adds 1 to the specified variable

```
addl    VAR
```

subl subtracts 1 from the specified variable

```
subl    VAR
```

zero sets to zero a single variable or a set of consecutive variables

```
zero    VAR
zero    STARTING VAR,NUM VARS
```

set sets values of consecutive variables starting at the specified variable, or sets values of matrix elements starting at the specified element (starts at the first element if no element is specified); can be used to set segmented arrays but not segmented variables

```
set     STARTING VAR← VALUE1,VALUE2,VALUE3,...
set     ARRAYNAME← VALUE1,VALUE2,VALUE3,...
set     ARRAYNAME(ROW,COLUMN)← VALUE1,VALUE2,VALUE3,...
```

Note: Up to 61 values may be set with a single -set- command.

Operations and symbols used in calculations

addition +

subtraction -

multiplication × or * (implied multiplication is permitted, e.g., 5a)

division ÷ or /

dot product of two arrays °

cross product of two arrays X

parentheses, brackets ( ), [ ], { }

exponentiation ** or superscript or shift-superscript (e.g., a^4)

assignment of a value to a variable ←

π = pi = 3.14159...

° = degree sign; indicates a number is interpreted in degrees, e.g., 30°;
number × 1° converts number to radians;
number ÷ 1° converts number to degrees

Address of a variable may be an expression; i.e., v(EXPR) is permitted, where EXPR is rounded to the nearest integer.

Precedence of operations (in brief)

operations within parentheses

exponentiation

multiplication

division

addition and subtraction

In general with anything but very simple expressions, parentheses should be used freely.

NOTE: The computer has approximately 14-digit accuracy.
Values of v-variables may range from about $\pm 10^{-293}$ to $\pm 10^{+322}$.
Values of n-variables may range from about -10^{17} to $+10^{17}$.
However, multiplication and division of large integer values may give erroneous results because of limitations on integer arithmetic.

System functions (argument may be an expression)

abs(x)	absolute value of x
arctan(x)	inverse tangent, result in radians, range $-\pi/2$ to $+\pi/2$; for result in degrees, use arctan(x)/1°
cos(x)	cosine of x, x in radians; use cos(x°) when x is in degrees
exp(x)	e^x
frac(x)	fractional part of x
int(x)	integer part of x

x is first rounded to the nearest integer if x is within 10^{-9} of the integer for $|x|<100$ or within $(10^{-11}\times|x|)$ of the integer for $|x|>100$ (approximately)

log(x)	common logarithm of x (base 10)
ln(x)	natural logarithm of x (base e)
round(x)	rounded value of x
sign(x)	= -1 for $x < -10^{-9}$; = 0 for $-10^{-9} \le x \le 10^{-9}$; = +1 for $x > 10^{-9}$
sin(x)	sine of x, x in radians; use sin(x°) when x is in degrees
sqrt(x)	square root of x
varloc(x)	address of variable x (x may be a student variable, central memory variable, or defined variable)

Logical operations and functions (logical "true" is -1; logical "false" is 0)

x = y	equal to
x ≠ y	not equal to
x ≤ y	less than or equal to
x ≥ y	greater than or equal to
x < y	less than
x > y	greater than

equality is "true" if $|x-y|<10^{-9}$ for $|x|<100$ or if $|x-y|<(10^{-11}\times|x|)$ for $|x|>100$ (approximately)

xandy	logical "and"; result is "true" only if both x and y are "true"
xory	logical "or"; result is "true" if either x or y or both are "true"
not(x)	reverse of truth value of x: if x=0, not(x)=-1; if x≠0, not(x)=0

Bit operations and functions (use with n-variables)

xarsy	shifts x to the right by y bit positions
xclsy	shifts x to the left (circularly) by y bit positions
x$mask$y	sets bits where bits are set in both x and y
x$union$y	sets bits where bits are set in either x or y or both
x$diff$y	sets bits where bits are set in either x or y but not both
bitcnt(x)	number of bits set in x
comp(x)	complement of x (bit reversal)
lmask(x)	left-justified octal number of x bits
rmask(x)	right-justified octal number of x bits

x ranges from 0 to "zbpw"

System functions continued on next page.

Array functions

And(x)	"true" (=-1) if all elements of array x are "true"
Max(x)	largest element in array x
Min(x)	smallest element in array x
Or(x)	"true" (=-1) if any element of array x is "true"
Prod(x)	product of all elements in array x
Rev(x)	reverse of array x; i.e., last element is now first, etc.
Sum(x)	sum of all elements in array x
Transp(x)	transpose of array x; i.e., rows and columns are interchanged

NOTE: Because of the finite accuracy of any computer, rounding occurs with operations with fractional values (v-variables), giving results which may be off by only one or two bits but which can lead to serious errors. The tolerances indicated with certain functions and logical operations are designed to avoid such problems by ignoring these least significant bits. However, there is no general solution to this inherent problem, and users must design checks for specific applications.

Random numbers

seed — specifies a seed for generation of random numbers with -randu- and -randp-; remains in effect until execution of another -seed- command

```
seed    VAR CONTAINING THE SEED VALUE
seed    (B)   (clears former value; specifies normal system seed)
```

randu — selects a random number, sampled with replacement, and places it in the specified variable

```
randu   VAR,MAXIMUM   (selects integer from 1 to MAXIMUM; MAXIMUM>Ø)
randu   VAR   (selects number from Ø to 1; if the tag is an
              n-variable, a value Ø or 1 is returned)
```

NOTE: The next four commands are generally used together to provide random numbers without replacement.

setperm — single-argument -setperm- sets up two lists of integers from 1 to the specified value for sampling without replacement, one a working copy of the list (affected by -randp-) and the other an inactive but alterable copy of the list (affected by -remove-); two-argument -setperm- sets up one list only; a separate copy of the list should be maintained for use with -remove-

```
setperm MAXIMUM INTEGER IN LIST  (Ø≤MAXIMUM≤12Ø)
setperm LIST LENGTH,STARTING VAR CONTAINING ELEMENTS IN LIST
          (for list length > 12Ø and for special purpose sampling;
          number of variables required is: 2+int[(length-1)/6Ø];
          length of the list is from Ø to 3ØØØ)
```

randp — selects a random integer without replacement from the working copy of the list set up by -setperm- and places it in the specified variable

```
randp   VAR   (with 1-argument -setperm-)
randp   VAR FOR STORING VALUE,STARTING VAR CONTAINING ELEMENTS
        IN LIST  (with 2-argument -setperm- or calculated list)
```

Note: With either form of -randp- the value returned is Ø if the working copy of the list is exhausted.

remove — removes the specified value from the inactive copy of the list set up by single-argument -setperm- or from the copy of the list for two-argument -remove-

```
remove  INTEGER TO BE REMOVED   (with 1-argument -setperm-)
remove  INTEGER TO BE REMOVED,STARTING VAR OF COPY OF LIST
          (with 2-argument -setperm-)
```

modperm (no tag) replaces the working copy of the list with the current copy of the inactive list from which integers may have been removed by -remove-; paired with single-argument -setperm- only; to simulate -modperm- with two-argument -setperm- use -block- or -transfr- to replace the working list with the copy of the list

Information

clock puts a character string in the specified variable giving time on a 24-hour clock in the format (space)hour.minute.second. (see also the system variable "clock")

clock VAR WHERE STRING IS STORED (use -showa- to display)

date puts a character string in the specified variable for the current date in the format (space)month number/day/last two digits of year(space)

date VAR WHERE STRING IS STORED (use -showa- to display)

day places in the specified variable the number of days elapsed since midnight December 31, 1972 to the nearest 10^{-6} day (approximately .1 second)

day VAR WHERE VALUE IS STORED (use a v-variable)

name places the sign-on name of the user (up to 18 characters) left-justified in two consecutive variables starting at the specified variable with octal zero fill in unused positions

name STARTING VAR (requires two consecutive variables)

course places the sign-on course of the user left-justifed in the specified variable, up to 8 characters with octal zero fill in unused positions

course VAR

compute compiles the specified character string, executes, and stores the result in the specified variable; sets a pointer to the compiled code

compute VAR WHERE RESULT IS STORED,STARTING VAR OF STRING,NUM CHARACTERS IN STRING,VAR FOR POINTER TO COMPILED CODE
(invalid expression does not compile; once compiled, no recompilation is done unless pointer is zeroed)

Note: Character string may be stored with -storea- or -pack- and may contain up to 100 characters.

Bit and character manipulation

search scans a character buffer for a specified object character string

search ARG1,ARG2,ARG3,ARG4,ARG5,ARG6

ARG1 = object string (left-justified)
ARG2 = number of characters in string (≤1Ø characters)
ARG3 = starting variable of buffer to be searched
ARG4 = total number of characters in buffer
ARG5 = relative character position at which to start search
ARG6 = variable for storing relative character position of first occurrence of object string

search ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7

ARG1 = object string (left-justified)
ARG2 = number of characters in string (≤1Ø characters)
ARG3 = starting variable of buffer to be searched
ARG4 = number of characters in buffer
ARG5 = relative character position at which to start search
ARG6 = variable for storing found count
ARG7 = number of variables following ARG6 for storing relative character positions of object string

Note: In both versions of -search- the relative found location is -1 if the object string is not found. In the second version the count is Ø if the string is not found. Relative position of the first character is 1, of the second character, 2, etc. If ARG4 (length of buffer) is negative, a backwards search is done.

pack packs a character string starting in the specified variable; the string will be left-justified with octal zero fill in unused positions; if the character count is not desired, the field is left blank

pack STARTING VAR FOR STORING STRING,VAR FOR STORING CHARACTER COUNT,STRING
pack STARTING VAR FOR STORING STRING,,STRING

Note: Use n-variable(s) for packing the string if subsequent comparison for equality with another string is done. (If the character string is packed for other purposes, v-variables are acceptable.) Segmented variables cannot be used with -pack-.
Embedded -show- (and related embeds) may be included in the string. Up to 5ØØ characters may be packed.

packc packs one of several character strings into a variable, depending on the rounded value of a conditional expression; the string will be left-justified with octal zero fill in unused positions; if the character count is not desired, the field may be left blank; blank argument for the character string leaves the variable(s) unchanged for that value of the conditional expression

packc EXPR↕STARTING VAR FOR STORING STRING↕VAR FOR STORING CHARACTER COUNT↕STRING↑↕STRING0↕STRING1↕↕STRING3...

Note: Up to 100 character strings may be listed. The character count is limited by the length of the line of code. (See -pack- for other options and restrictions.)

itoa converts an integer to a character string, left-justified with octal zero fill in unused positions

itoa VAR WHERE INTEGER IS STORED,STARTING VAR FOR STORING STRING,VAR FOR STORING NUM CHARACTERS (opt)

Note: Non-integer values are rounded to the nearest integer before conversion to a character string.

move moves character(s) from a specified character position in a character string to a specified position in another character string

move FROM STARTING VAR,FROM STARTING POSITION,TO STARTING VAR,TO STARTING POSITION,NUM CHARACTERS (opt)
move 'STRING',FROM STARTING POSITION,TO STARTING VAR,TO STARTING POSITION,NUM CHARACTERS (opt)

Note: If not specified, number of characters is 1. Maximum number of characters is 5000. Positions may be >10.

otoa converts a number from octal format to alphanumeric format (i.e., to a character string) for the number of octal digits, if given (digits are counted from the right end of the number)

otoa NUMBER,STARTING VAR FOR STORING STRING,NUM DIGITS (opt)

Note: Number of digits, if omitted, is 20.
If number of digits ≤10, 1 variable is used for storing the string; otherwise 2 variables are used.

htoa similar to -otoa- but for hexadecimal to alphanumeric conversion

htoa NUMBER,STARTING VAR FOR STORING STRING,NUM DIGITS (opt)

Note: Number of digits, if omitted, is 15.
If number of digits ≤10, 1 variable is used for storing the string; otherwise 2 variables are used.

Operations on lists

sort arranges a list of entries stored in consecutive variables in ascending order according to the value of the specified sort field

```
sort    ARG1;ARG2,ARG3,ARG4,ARG5,ARG6
        ARG1A;ARG2A   (optional line; allows simultaneous sorting
           of an associated list of entries)

        ARG1 = starting location; may be:
               STUDENT VAR (v or n)  or
               CM VAR (vc or nc)  or
               c,ECS COMMON LOCATION  or
               s,ECS STORAGE LOCATION
        ARG2 = number of entries in list
        ARG3 = number of variables per entry (or increment between
               entries); value from 1 to 200
        ARG4 = starting bit position of sort field
        ARG5 = number of bits in sort field
        ARG6 = mask on sort field (optional)
        ARG1A = starting location (see ARG1 for details) (optional)
        ARG2A = number of variables per entry (optional)
```

Note: The field for numerical sorting may not extend across boundaries of variables.

sorta arranges a list of entries stored in consecutive variables in alphabetical order according to the internal codes for the characters in the specified sort field

```
sorta   ARG1;ARG2,ARG3,ARG4,ARG5,ARG6
        ARG1A;ARG2A   (optional line; allows simultaneous sorting
           of an associated list of entries)

        ARG1 = starting location; may be:
               STUDENT VAR (v or n)  or
               CM VAR (vc or nc)  or
               c,ECS COMMON LOCATION  or
               s,ECS STORAGE LOCATION
        ARG2 = number of entries in list
        ARG3 = number of variables per entry (or increment between
               entries); value from 1 to 200
        ARG4 = starting character position of sort field
        ARG5 = number of characters in sort field
        ARG6 = mask on sort field (optional)
        ARG1A = starting location (see ARG1 for details) (optional)
        ARG2A = number of variables per entry (optional)
```

Note: The field for alphabetical sorting may extend across boundaries of variables. However, the mask can affect only one variable.

finds performs a binary chop search on a list sorted with -sort- command

finds ARG1,ARG2;ARG3,ARG4,ARG5,ARG6,ARG7,ARG8

ARG1 = object of search
ARG2 = starting location of list; may be:
STUDENT VAR (v or n) or
CM VAR (vc or nc) or
c,ECS COMMON LOCATION or
s,ECS STORAGE LOCATION
ARG3 = number of entries in list
ARG4 = number of variables per entry (or increment between entries); value from 1 to 500
ARG5 = starting bit position of sort field
ARG6 = number of bits in sort field
ARG7 = variable for storing relative value of found location (1st entry is 1, 2nd is 2, etc); if object is not found, variable is set to negative of position where object should have been found
ARG8 = mask on sort field (optional)

findsa performs a binary chop search on a list sorted with -sorta- command

findsa ARG1,ARG2;ARG3,ARG4,ARG5,ARG6,ARG7,ARG8

ARG1 = object of search
ARG2 = starting location of list; may be:
STUDENT VAR (v or n) or
CM VAR (vc or nc) or
c,ECS COMMON LOCATION or
s,ECS STORAGE LOCATION
ARG3 = number of entries in list
ARG4 = number of variables per entry (or increment between entries); value from 1 to 500
ARG5 = starting character position of sort field
ARG6 = number of characters in sort field
ARG7 = variable for storing relative value of found location (1st entry is 1, 2nd is 2, etc); if object is not found, variable is set to negative of position where object should have been found
ARG8 = mask on sort field (optional)

inserts inserts contents of specified buffer into a list of entries stored in consecutive variables; shifts the remainder of the list down

```
inserts ARG1,ARG2;ARG3,ARG4,ARG5,ARG6
        ARG1A,ARG2A;ARG3A   (optional line; allows simultaneous
           insertion into associated list)

        ARG1 = variable containing object to be inserted
        ARG2 = starting location of list; may be:
               STUDENT VAR (v or n)  or
               CM VAR (vc or nc)  or
               c,ECS COMMON LOCATION  or
               s,ECS STORAGE LOCATION
        ARG3 = number of entries in list
        ARG4 = number of variables per entry (or increment between
               entries); ARG4 × ARG6 = value from 1 to 500
        ARG5 = relative position in list at which to insert object
               (must be value from 1 to 1+length of list)
        ARG6 = number of entries to insert (optional; default is 1)
        ARG1A = variable containing object to be inserted (optional)
        ARG2A = starting location (see ARG2 for details) (optional)
        ARG3A = number of variables per entry (optional)
```

deletes deletes the entry at the specified position in a list of entries stored in consecutive variables; shifts the remainder of the list up and fills the last entry with zeros

```
deletes ARG1;ARG2,ARG3,ARG4
        ARG1A;ARG2A   (optional line; allows simultaneous deletion
           from an associated list)

        ARG1 = starting location; may be:
               STUDENT VAR (v or n)  or
               CM VAR (vc or nc)  or
               c,ECS COMMON LOCATION  or
               s,ECS STORAGE LOCATION
        ARG2 = number of entries in list
        ARG3 = number of variables per entry (or increment between
               entries); value from 1 to 500
        ARG4 = relative position in the list of the entry to be
               deleted (must be a value from 1 to length of list)
        ARG5 = number of entries to delete (optional; default is 1)
        ARG1A = starting variable (see ARG1 for details) (optional)
        ARG2A = number of variables per entry (optional)
```

NOTE: Commands -inserts- and -deletes- may be used on both sorted and unsorted lists.

find scans each variable in a set of consecutive variables for the first variable containing the specified bit pattern

find ARG1,ARG2,ARG3,ARG4,ARG5,ARG6

ARG1 = object bit pattern
ARG2 = starting variable in list
ARG3 = number of variables in list
ARG4 = variable for storing relative found location
ARG5 = increment between variables (optional)
ARG6 = mask (optional)

findall scans each variable in a set of consecutive variables for all variables containing the specified bit pattern

findall ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7

ARG1 = object bit pattern
ARG2 = starting variable in list
ARG3 = number of variables in list
ARG4 = variable for storing total found count
ARG5 = number of following variables for storing relative found locations (may be less than total count)
ARG6 = increment between variables (optional)
ARG7 = mask (optional)

NOTE: Use n-variables with -find- and -findall-. Segmented variables may not be used. Increment, if omitted, is 1. Negative increment causes a backward scan from the last variable in the list.
If the mask is omitted, the entire variable is compared with the object bit pattern. If a mask is used, the increment must be given, even if it is 1.
Relative position of the first variable is Ø, of the second variable, 1, etc.
With -find-, if the bit pattern is not found, the found location is -1.
With -findall-, if the bit pattern is not found, the count is Ø and the first following variable is -1.

Data manipulation

block copies a set of consecutive student variables (v, n) or central memory variables (vc, nc) into another set of consecutive variables

block FROM STARTING VAR,TO STARTING VAR,NUM VARS

transfr transfers data between v- and n-variables (student variables), vc- and nc-variables (central memory variables), ECS common, ECS storage, or vr- and nr-variables (router variables) (-comload- and/or -stoload- must be in effect with central memory variables)

transfr FROM STARTING LOCATION;TO STARTING LOCATION;LENGTH
(general form)

Any of the following may be used in the first two arguments of the tag:

STUDENT VAR (v or n)
common,ECS COMMON LOCATION or c,ECS COMMON LOCATION
storage,ECS STORAGE LOCATION or s,ECS STORAGE LOCATION
CM VAR (vc or nc)
router,ECS ROUTER COMMON LOCATION (when placed in "from" position, router lesson must contain -allow read-; when placed in "to" position, router lesson must contain -allow write-)
routvars,ECS ROUTER VAR LOCATION (may be placed only in "from" position; router lesson must contain -allow read rvars-)
ROUTER VAR (legal only when -transfr- command is in the router lesson)

for example:

transfr v1;c,23;1Ø (transfers variables v1 through v1Ø to ECS common locations 23 through 32)
transfr v6;vc51;9 (transfers variables v6 through v14 to variables vc51 through vc59)
transfr s,11;c,1ØØ;21 (transfers ECS storage locations 11 through 31 to ECS common locations 1ØØ through 12Ø)

Note: Limit to length which may be transferred is set by:
tag of -common- or -storage- (when referencing ECS) or
length of -comload- or -stoload- (when referencing CM variables) or
15Ø (when referencing student variables) or
tag of -routvar- (when referencing router variables)

NOTE: For this type of operation, -block- and -transfr- are very fast.

common (non-executable) sets up storage space which is accessible to all students in a lesson; in central memory the variables are referenced by vc and nc; common codewords must match when the common blocks are in a different lesson from the -common- command

```
common  LENGTH OF TEMPORARY COMMON   (variable tag is not permitted)
common  ,COMMON NAME,LENGTH OF PERMANENT COMMON
common  LESSON NAME,COMMON NAME,LENGTH OF PERMANENT COMMON
           (LESSON NAME is the lesson containing the common blocks)
```

Note: Maximum length is 8ØØØ words. For length ≤ 15ØØ, loading and unloading are automatic unless altered by -comload- or optional tags "no load" or "read only".
For length > 15ØØ, -comload- must be used for access to central memory variables. Either or both of the following optional arguments may be added to the tag of -common-:

```
no load    (cancels automatic loading of common from ECS to CM)
read only  (prevents transfer of common from CM to ECS)
```

The following optional argument may be used only with permanent common:

```
checkpt    (causes common to be returned to disk approximately
            every 8 minutes)
```

comload provides automatic loading and unloading of common between central memory and ECS during each time slice; required if common length exceeds 15ØØ and central memory variables must be accessed

```
comload STARTING CM VAR (vc or nc),ECS COMMON LOCATION,NUM VARS
          (maximum of 15ØØ variables)
comload (B)   (unloads CM variables and turns off further automatic
   loading until another -comload- command is executed)
```

for example:

```
comload vc22,1Ø,8   (transfers vc22 through vc29 from and to
                     ECS locations 1Ø through 17)
```

comret (no tag) returns to disk the common specified by the lesson

```
Note:  zreturn = -1  if common was successfully returned
               =  Ø  if no common is specified for this lesson
               = +1  if common cannot be returned
```

abort prevents return of information to disk

```
abort   common
abort   record      (with student record; sets "user" to "sabort")
abort   autocheck   (with student record; sets "user" to "snockpt")
```

commonx similar to -common- except that -commonx- is executable; when -commonx- and common blocks are in different lessons, either the common codewords must match or the codeword argument must be given; if optional arguments (see -common-) are included, the codeword argument or the field must be present; only one common (determined by -common- or -commonx-) is permitted in a lesson

```
commonx ,COMMON NAME,LENGTH
commonx ,COMMON NAME,LENGTH,,OPTIONS (opt)
commonx LESSON NAME,COMMON NAME,LENGTH,'CODEWORD'(opt),OPTIONS(opt)
```

Note: Variable arguments must be enclosed in parentheses, and quote marks on the codeword are omitted.

```
zreturn = -1  if execution was successful
        =  Ø  if the common was not found
        =  1  if no codewords match
        =  2  if the lesson already has a common
        =  3  if the ECS version of the common has a
              different length
        =  4  if the length of the common is illegal
```

initial specifies a unit to be executed immediately by the first user to encounter this command when a lesson or common is first brought into ECS

```
initial common,UNIT NAME   (not executed if common is in another
          lesson which is already in ECS and in which
          -initial  common- has already been executed)
initial lesson,UNIT NAME
```

storage (non-executable) sets up storage space which serves as a temporary extension of student variables; in central memory the variables are referenced by vc and nc; they are not saved in student records and are zeroed during jumpout to another lesson (for exception see -inhibit dropstor-)

```
storage LENGTH OF STORAGE   (maximum length of 15ØØ)
```

stoload similar to -comload- but refers to storage; required for any length storage to access central memory variables

```
stoload STARTING CM VAR (vc or nc),ECS STORAGE LOCATION,NUM VARS
          (maximum of 15ØØ variables)
stoload (B)   (turns off automatic loading and unloading of storage)
```

NOTE: When both -comload- and -stoload- are used, care must be exercised that the addresses of the central memory variables into which common has been loaded do not overlap with the addresses into which storage has been loaded.

dataset establishes a connection with a special type of file which permits disk records to be read and written in; record size may be from 64 words to 512 words; if the records are grouped by name, the file is called a "nameset"

dataset NAME (common codewords on lesson and on file must match)
dataset NAME,EXPR,'CODEWORD' (opt) (for read and write access: EXPR=-1, CODEWORD is file common codeword [if CODEWORD is omitted, common codewords on lesson and on file must match]; for read only access: EXPR≠-1, CODEWORD is either file common codeword or inspect codeword [if CODEWORD is omitted, common codeword on lesson must match either file common or inspect codeword])
dataset (B) (disconnects and releases current file)

Note: Variable first or third argument must be enclosed in parentheses, and quote marks on third argument are omitted.

zreturn = -1 if connection to the file is successful
= Ø if no such dataset or nameset exists
= 1 if no codewords match
= 2 if the file directory is being edited
≥ 3 if disk errors occur

NOTE: When -datain-, -dataout-, -release-, -reserve- are used with namesets, the record designations are relative to the selected named records.

datain transfers data from disk to the desired buffer

datain STARTING RECORD NUMBER;TO STARTING LOCATION;NUM RECORDS (opt)

dataout transfers data from a buffer to disk

dataout STARTING RECORD NUMBER;FROM STARTING LOCATION;NUM RECORDS (opt)

NOTE: With -datain- and -dataout- the second argument in the tag may be: STUDENT VAR or c,ECS COMMON LOCATION or s,ECS STORAGE LOCATION The number of records transferred cannot exceed the capacity of the buffer. If omitted, the number of records transferred is 1.

zreturn = -1 if -datain- or -dataout- is executed successfully
= Ø if there is a pack error
= 1 if there is a file error or if there is no preceding successful -dataset- command
= 2 if record numbers extend out of range
= 3 if the required buffer locations extend out of range
= 4 (-dataout- only) if write permission has not been granted
= 5 (-dataout- only) if record(s) is reserved
≥ 6 if there are disk errors

reserve sets system variable "zreturn" in order to allow a user to reserve the common or specified dataset or nameset records to prevent changes by more than one user at a time

reserve common

Note: zreturn = -2 if the common is already reserved by this user
= -1 if -reserve- is executed successfully by this user
= station number of user who has already reserved the common

reserve dataset,STARTING RECORD NUMBER,NUM RECORDS
reserve dataset (reserves all records in the dataset or nameset)

Note: zreturn = -2 if these records are reserved by this user
= -1 if -reserve- is executed successfully by this user
= Ø if there is no preceding successful -dataset- command
= 1 if record numbers extend out of range
= 2 if read-only access is allowed
= 5+station number of user who has reserved these records

release sets system variable "zreturn" to allow the common or specified dataset or nameset records to be released

release common

Note: zreturn = -2 if the common is not reserved by any user
= -1 if -release- is executed successfully by this user
= station number of user who has reserved the common

release dataset,STARTING RECORD NUMBER,NUM RECORDS
release dataset (releases all records in the daset or nameset)

Note: zreturn = -2 if these records are not reserved by any user
= -1 if -release- is executed successfully by this user
= Ø if there is no preceding successful -dataset- command
= 1 if record numbers extend out of range
= 5+station number of user who has reserved these records

NOTE: With the nameset commands (-setname-, -getname-, -addname-, -rename-, -addrecs-, -delrecs-, -delname-, -names-), the name can be up to 30 characters long (3 variables). The optional extra information for the name is stored in the right-most 24 bits of the specified variable.

setname selects a name (i.e., named set of records) in a nameset

```
setname STARTING VAR CONTAINING NAME
setname nextname   (selects the next name in alphabetical order or
          the first name if a name has not already been selected)
setname backname   (selects the preceding name in alphabetical order
          or the last name if a name has not already been selected)
setname (B)   (clears the name currently selected)

Note:  zreturn = -1  if the specified name matches exactly a name
                     in the nameset
               =  0  if the specified name matches one name for the
                     given number of characters; selects that name
               =  1  if the specified name matches more than one
                     name for the given number of characters;
                     selects the first of the names
               =  2  if the specified name does not match any name;
                     the name reference is cleared
               =  3  if no nameset is in effect

(With tags "nextname" and "backname", "zreturn" can have only
values -1, 2, or 3.)
```

getname stores the name currently selected and the associated information, if specified; name is left-justified and unused character positions are filled with octal zeros; extra information is stored in right-most 24 bits of the specified variable and remaining bits are cleared

```
getname STARTING VAR FOR STORING NAME,VAR FOR STORING EXTRA
        INFORMATION (opt)

Note:  If no name has been selected, a value of 0 is stored.
```

addname adds a new named set of records to a nameset file; selects that name

```
addname STARTING VAR CONTAINING NAME,NUM RECORDS (opt),VAR
        CONTAINING EXTRA INFORMATION (opt)

Note:  Number of records, if omitted, is 1.
       Extra information, if omitted, is 0.

       zreturn = -1  if addition of name was successful
               =  0  if no nameset is in effect
               =  1  if user is not allowed write access
               =  2  if new name duplicates an existing name
               =  3  if enough room is not available for new records
               =  4  if the nameset is reserved
```

rename changes the name of the currently selected named set of records and/or the associated extra information

rename STARTING VAR CONTAINING NEW NAME,VAR CONTAINING NEW INFORMATION (opt)

Note: If the second argument is omitted, information is unchanged.

zreturn = -1 if name was changed successfully
= Ø if no nameset is in effect
= 1 if user does not have write access
= 2 if no name has been referenced
= 3 if new name duplicates an existing name
= 4 if records in the selected name are reserved

addrecs adds records to the selected name

addrecs NUM RECORDS TO ADD AT END
addrecs RECORD POSITION,NUM RECORDS TO ADD AT THAT POSITION

Note: zreturn = -1 if records were added successfully
= Ø if no nameset is in effect
= 1 if user does not have write access
= 2 if no name has been selected
= 3 if the specified starting position is outside the range of records in the named records (2-argument form)
= 4 if records in the selected name are reserved
= 5 if enough room is not available for new records

delrecs deletes records from the selected name

delrecs NUM RECORDS TO DELETE FROM END
delrecs RECORD POSITION,NUM RECORDS TO DELETE STARTING AT THAT POSITION

Note: zreturn = -1 if records were deleted successfully
= Ø if no nameset is in effect
= 1 if user does not have write access
= 2 if no name has been selected
= 3 if the specified starting position is outside the range of records in the named records (2-argument form)
= 4 if the specified records are reserved

delname (no tag) deletes the currently selected name and all its records

Note: zreturn = -1 if records were deleted successfully
= Ø if no nameset is in effect
= 1 if user does not have write access
= 2 if no name has been selected
= 4 if the specified records are reserved

names stores a set of names from the list in the nameset; each name entry, which may require from 1 to 3 variables, is followed by a variable whose left-most 15 bits contain the number of associated records and whose right-most 24 bits contain the extra information

names ARG1,ARG2,ARG3,ARG4

ARG1 = starting postion in list of nameset names
ARG2 = starting variable for storing names
ARG3 = maximum number of variables for storing names (each requires from 2 to 4 variables)
ARG4 = variable for storing actual number of names obtained

Note: zreturn = -1 if names were stored successfully
= Ø if no nameset is in effect
= +1 if the starting position is invalid

backgnd (no tag) flags a lesson as a "background" lesson so that it may use large amounts of CPU time when the time is available; when CPU time is scarce, the lesson is handled at lower priority than non-background lessons

foregnd (no tag) switches the lesson to foreground processing; normal state of execution

System variables for calculating

lcommon length of common (set by tag of -common- command)

lstorag length of storage (set by tag of -storage- command)

zbpc number of bits per character (currently 6)

zbpw number of bits per computer word (currently 6Ø)

zcpw number of characters per computer word (currently 1Ø)

zcusers number of users signed into the current common

zdsname name of the dataset or nameset currently attached to the lesson (left-justified; display with -showa-)

zdusers number of users connected to the current dataset (=Ø if no dataset is attached)

zinfo contains the 24 bits of information associated with the currently selected name in a nameset; stored in the right-most 24 bits

znscpn number of characters per name for the attached nameset (=Ø if no nameset is attached, including the case where a dataset is attached)

znsmaxn maximum number of names allowed in the attached nameset

znsmaxr maximum number of records allowed in the attached nameset

znsnams number of names currently in use in the attached nameset

znsrecs number of records currently in the entire nameset

zrecs number of records in the currently selected name or
number of records in the dataset currently attached
(=Ø if no dataset is attached or if no name has been selected)

zwpr number of computer words per record in the attached dataset or in the attached nameset (=Ø if no dataset or nameset is attached)

Requesting data

dataon specifies that student data for the lesson is to be collected and sent to a datafile if the student records have been set to allow collection of data (see also system variable "dataon")

```
dataon  (B)   (turns on all data collection)
dataon  TAG   (TAG can be ok, no, unrec no, vocab, area, output,
              help, help no, term, term no, errors, signin)
```

Note: Non-blank tag will temporarily override options set in course records until turned off by -dataoff- with appropriate tag.

dataoff specifies termination of collection of data for that lesson

```
dataoff (B)
dataoff TAG  (TAG is same as that used in a preceding -dataon-)
```

Note: Non-blank tag of -dataoff- will turn off only options turned on by a previous -dataon- with a non-blank tag; -dataoff- does not override options set in course records.

Classifying data

area — specifies a section in the lesson (called an area) typically representing 5 to 15 minutes of student contact time for which certain data is collected

area NAME (maximum of 10 characters in NAME; cannot start with a number; variable tag must be enclosed in parentheses)

Note: Information collected:

elapsed time in the area (in minutes; accurate to .1 minute)
number of arrows encountered
number of "ok" judgments
number of responses judged "ok" on 1st attempt
number of anticipated "no" judgments, or "wrong" judgments
number of unanticipated "no" judgments
number of term requests satisfied
number of term requests not satisfied
number of help-type requests satisfied
number of help-type requests not satisfied
whether the area has been completed

area (B) (causes data for the preceding area to be placed in the datafile; no further data is stored until -area- with nonblank tag is executed)
area incomplete (terminates the current area and flags it as incomplete)
area cancelled (cancels all data for the current area; does not initiate a new area or produce any records in the datafile)

output — puts a comment and/or value of an expression into the datafile

output COMMENT AND/OR EXPR (formats for the expression are: a, n, o, v with embedded form, i.e., ≺ FORMAT,EXPR≻)

outputl — places labeled information from specified student variables in the datafile

outputl LABEL (opt),STARTING VAR,NUM VARS (maximum of 10 characters in LABEL and 20 consecutive variables)

setdat — allows alteration of system variables containing area data

setdat SYSTEM VAR⇐ EXPR

Note: "atime" cannot be set to a value greater than the total time signed on for that session. It is accurate to 0.1 second. The remaining system variables (except for "aarea") can be set to values up to 511.

Transferring data

readset establishes a link between a datafile or course and the lesson which will receive the data

```
readset COURSE NAME,'COURSE CHANGE OR INSPECT CODEWORD' (opt),
        VAR FOR STORING NUM RECORDS (opt)
readset DATAFILE NAME,'DATAFILE CHANGE OR INSPECT CODEWORD'(opt)
        VAR FOR STORING NUM UNUSED BLOCKS (opt)
```

Note: The second argument is included if codewords on the lesson and datafile or course do not match.
Variable arguments must be enclosed in parentheses; if the codeword is a variable, quote marks are omitted.
For a datafile, the third argument is -1 if the datafile is full.

```
zreturn = -2  if the course exists and is not empty
        = -1  if the datafile exists and is not empty
        =  Ø  if the name specified is not a datafile or
              a course
        =  1  if codewords on the lesson and datafile or
              course do not match
        =  2  if the datafile or course is empty
        =  3  if there is no room in ECS for the buffer
        =  4  if there is a disk error
```

readd transfers data from a datafile into student variables or central memory variables (must be preceded by -readset- naming the datafile)

```
readd   area,STARTING VAR,NUM VARS
```

Note: Area summary data consists of the following information:

```
n(x) or nc(x) = starting variable
n(x) and n(x+1) contain the sign-on name (up to 18 characters)
n(x+2) contains the lesson name
n(x+3)      "   the area name
n(x+4)      "   elapsed time for the area (in milliseconds)
n(x+5)      "   number of arrows for the area
n(x+6)      "   number of "ok" judgments for the area
n(x+7)      "   number of "ok" judgments on the 1st attempt
n(x+8)      "   number of anticipated "no" judgments (matched by
                -wrong-, -wrongc-, -wrongu-, -wrongv-, -miscon-,
                -touchw-, or -judge  wrong-)
n(x+9)      "   number of unanticipated "no" judgments
n(x+1Ø)     "   number of help-type requests satisfied
n(x+11)     "   number of help-type requests not satisfied
n(x+12)     "   number of term requests satisfied
n(x+13)     "   number of term requests not satisfied
n(x+14) = -1 if the area was completed, =Ø if not
```

(-readd- continued on next page)

```
readd   output1,STARTING VAR,NUM VARS

Note:  Data from -output1- consists of the following information:

n(x) or nc(x) = starting variable
n(x) contains the length of -output1- (number of variables)
n(x+1) and n(x+2) contain the sign-on name (up to 18 characters)
n(x+3) contains the lesson name
n(x+4)     "    the area name
n(x+5)     "    execution time of -output1- in milliseconds
n(x+6)     "    -output1- label
n(x+7) to n(x + NUM VARS - 1) contain data in the tag of -output1-

readd   signoff,STARTING VAR,NUM VARS

Note:  Signoff data consists of the following information:

n(x) or nc(x) = starting variable
n(x) and n(x+1) contain the sign-on name (up to 18 characters)
n(x+2) contains the lesson name
n(x+3)     "    elapsed time (in minutes) spent in the lesson
                during this session
n(x+4)     "    total time (in minutes) to complete the lesson
                if the lesson is completed during this session  or
                -1 if the lesson is not completed during this session
n(x+5)     "    date of session
n(x+6)     "    time of signoff


Note:  With all tags for -readd-,

       zreturn = -1  if there is more data in the datafile
               =  0  if the end of the datafile is reached
```

readr allows access to the course roster, to course records of the user whose name is in the specified variables, or to records of all users in sequence (must be preceded by -readset- naming the course)

```
readr   roster,STARTING POSITION IN ROSTER;TO STARTING VAR;
        NUM NAMES   (reads names only from the roster; each
           name requires two variables)

readr   name,STARTING VAR CONTAINING NAME
        stats,STARTING WORD;TO STARTING VAR;NUM VARS
           (reads course statistics)
        rvars,STARTING ADDRESS;TO STARTING VAR;NUM VARS
           (reads router variables)
        svars,STARTING ADDRESS;TO STARTING VAR;NUM VARS
           (reads student variables)
        ldone,STARTING LESSON POSITION;TO STARTING VAR;NUM VARS
           ("ldone" information, in 3-bit signed segments)
        lscore,STARTING LESSON POSITION;TO STARTING VAR;NUM VARS
           ("lscore" information, in 8-bit signed segments)

readr   sequential
        stats,STARTING WORD;TO STARTING VAR;NUM VARS
           (etc.  same options as for individual name; if
           preceded by -readr  name-, -readr  sequential-
           starts reading the roster at the next name; otherwise
           it starts reading at the beginning of the roster; use
           -readr  sequential- in a loop)

Note:  With "stats" option:

word 1 and word 2 contain the sign-on name (up to 18 characters)
                  (display with -showa-)
word 3 contains the user type (display with -showa-)
word 4     "    date record was created (display with -showa-)
word 5     "    date user last signed on (display with -showa-)
word 6     "    hour user last signed on (display with -showa-)
word 7     "    total time (in hours) on the system (display
                with -show-; use v-variable)
word 8     "    cpu usage in thousand instructions per second (tips)
                (display with -show-; use v-variable)
word 9     "    total number of days on the system (display
                with -show-)
word 10    "    number of sessions on the system (display
                with -show-)
word 11    "    cumulative number of disk accesses per minute (dapm)
                (display with -show-; use v-variable)

       zreturn = -1  if there is more data in the course
               =  0  if the last record has been read
               =  1  if the name is not in the course
               =  2  if the course is not using the system router
                     (with -readr  ldone-, -readr  lscore-)
```

Student sign-on and sign-off

restart specifies unit (and lesson if given) where the student is to begin on the next sign-on

```
restart (B)   (start at the main unit containing this command)
restart UNIT NAME   (start in this lesson at the specified unit)
restart LESSON NAME,UNIT NAME   (start in the specified lesson at
          the specified unit)
restart <LESLIST POSITION>,UNIT NAME   (start in the lesson
          at the specified leslist position, in the specified
          unit; variable unit names must be enclosed in
          parentheses)
```

finish specifies the unit which will be executed upon exit from the lesson via STOP1 (but not via -end lesson-)

```
finish  UNIT NAME
finish  (B)  or  finish  q   (clears -finish- setting)
finish  EXPR,NAME1,NAMEØ,q,NAME2,x   (example of conditional form;
          maximum of 1ØØ arguments in the conditional tag)
```

System variables for data keeping

collecting data

dataon = -1 if data collection is turned on
= Ø if data collection is off
(see also command -dataon-)

session data

zsesset elapsed time since sign-on during this session (in seconds, to nearest millisecond)

zsesspt processing time since sign-on during this session (in seconds, to nearest millisecond)

zsessda number of disk accesses since sign-on during this session

area data

aarea name of current area (left-justified; display with -showa-)

aarrows number of arrows encountered

ahelp number of help-type requests satisfied

ahelpn number of help-type requests not satisfied

aok number of "ok" judgments

aokist number of "ok" judgments on the first attempt

asno number of specified (anticipated) "no" judgments; also referred to as "wrong" judgments, where "judged" has been set to Ø

aterm number of term requests satisfied

atermn number of term requests not satisfied

atime elapsed time in the area (in milliseconds)

auno number of unanticipated "no" judgments; "judged" has been set to +1

NOTE: The system variables containing area data are zeroed at the beginning of each area.
"atime" may have a value up to about 9 hours.
The remaining variables (except "aarea") may have values up to 511.

Two types of commands are described in this section: judging commands and regular commands. (In all other sections of this book only regular commands are described.) Regular commands are not executed during the judging process, i.e., after the student has entered a response, nor are judging commands executed before the student has entered a response or in situations where no response is involved. (See The TUTOR Language and lesson "aids" for extensive descriptions of the judging process.)

Regular commands in this section include: -eraseu-, -force-, -edit-, -arrow-, -arrowa-, -arheada-, -long-, -jkey-, -copy-, -endarrow-, -getword-, -getmark-, -getloc-, -compare-, -iarrow-, -iarrowa-, -judge-, -okword-, -noword-, and -markup-. Commands -list-, -endings-, -vocabs-, and -vocab- are special non-executable commands which establish lists of words for use with certain response-matching commands. The -join- command is both regular and judging. The remaining commands in this section are judging commands.

Preparation for responding

eraseu (regular command) the specified unit is executed at all subsequent arrows in the unit containing -eraseu- when the student erases part or all of a response after receiving judgment; useful for erasing complicated displays which are not handled by the standard judging process

```
eraseu  UNIT NAME
eraseu  (B) or eraseu  q   (clears -eraseu- setting for remainder
           of the unit)
eraseu  EXPR,NAMEM,NAMEØ,q,NAME2,x   (example of conditional form;
           maximum of 1ØØ arguments in the conditional tag)
```

force (regular command) alters the input of a response as specified; setting is cleared at each main unit

```
force   firsterase   (erases an incorrect response and contingent
           message when the student presses any key, not just
           NEXT, ERASE, etc.)
force   font   (inserts the font code before the first keypress)
force   left   (forces the response to be written from right to left)
force   long   (initiates judging when the character input reaches
           the value of the tag of -long-; unnecessary with -long  1-)
force   micro  (forces all keypresses through the microtable
           conversion)
force   (B) or force   clear   (clears previous -force- settings
           in that unit)
force   clear,font,left   (may combine tags)
```

edit (regular command) required for EDIT key to be active when the tag of -long- exceeds 150; specifies the starting variable of a buffer for storing the characters in a response (up to 300 characters)

```
edit    STARTING VAR   (use student variable)
edit    (B)   (clears edit buffer; if placed after -arrow-, prevents
                use of the EDIT key; see also -inhibit  edit-)
```

arrow (regular command) plots the response arrow at the specified screen location (see -inhibit arrow-); sets defaults: -long 150- and -jkey (B)-

```
arrow   COARSE
arrow   FINEX,FINEY
```

arrowa (regular command) allows an alternative arrow associated with -iarrowa- and -arheada-; follows same rules and restrictions as -arrow-

```
arrowa  COARSE
arrowa  FINEX,FINEY
```

arheada (regular command) specifies a symbol to be plotted with the alternative arrow

```
arheada SYMBOL TO BE PLOTTED WITH -arrowa-
```

Note: Up to five 6-bit characters may be specified.

NOTE: To affect the first response, the following three commands (-long-, -jkey-, and -copy-) must follow the -arrow- command but must precede any judging commands. However, after the student enters a response (e.g., an incorrect response), these commands can be executed among the regular commands following the matched response in order to affect the next response at the same arrow.

long (regular command) sets the maximum number of characters in a response (default is 150 characters); must follow -arrow-

```
long    NUM CHARACTERS   (value of tag is from 0 to 300; -long  1-
           causes automatic judging; tag > 150 requires use of
           -edit- for EDIT key to be active; -long  0- prevents
           input from the keyset)
```

jkey (regular command) specifies the function key(s) which will initiate judging (in addition to the NEXT key); must follow -arrow-

```
jkey    KEYNAME   (name of key is in lower case)
jkey    KEYNAME1,KEYNAME2,KEYNAME3,...
jkey    (B)   (clears previous -jkey- settings so that only NEXT
          initiates judging)
```

copy (regular command) specifies the starting variable of the character string which is to be copied on the screen, one word at a time, when the COPY key is pressed; must follow -arrow-

```
copy    STARTING VAR,NUM CHARACTERS   (use student variable)
```

endarrow (regular command) (no tag) terminates judging with an unanticipated "no" judgment if the response has not been matched; after an "ok" judgment, -endarrow- terminates the search for additional -arrow- commands and switches back to the pre-arrow state

Vocabulary lists

list (non-executable) sets up a list of synonyms for judging; used with -answer-, -wrong-, -answerc-, -wrongc-, and -match-

```
list    LISTNAME,WORD1,PHRASE*CONSISTING*OF*SEVERAL*WORDS,
        WORD2,WORD3,...   (maximum of 7 characters in LISTNAME)
```

endings (non-executable) used with -vocabs- and -vocab- to add endings to root words (must precede -vocabs- or -vocab-)

```
endings NUMBER,ENDING1,ENDING2,...  (NUMBER is an integer from Ø to 9)
```

Note: In -vocabs- or -vocab-,

- WORD/NUMBER adds endings to the root word and includes all words in the vocabulary
- WORD//NUMBER adds only words with endings to the vocabulary; the root word is not included

Up to 1Ø -endings- commands with up to 8 endings each may be included. Apostrophe is legal in an ending.

vocabs (non-executable) sets up lists of ignorable words and synonymous important words; used with -concept- and -miscon-; does capitalization and spelling checks

```
vocabs  NAME
        <IGNORABLE WORDS SEPARATED BY COMMAS>
        WORD1,WORD2,PHRASE*CONSISTING*OF*SEVERAL*WORDS
        (SYNONYMOUS WORDS3 AND PHRASES SEPARATED BY COMMAS)
        (WORD4/s,WORD5/ENDING1/ENDING2,WORD6//ENDING1)
        WORD7/NUMBER1,WORD8//NUMBER2
        (WORD9=1,WORD1Ø=2,SYNONYM1Ø=2,...)
```

vocab (non-executable) similar to -vocabs- except does not check for capitalization and spelling and does not allow phrases

```
vocab   NAME
        <IGNORABLE WORDS SEPARATED BY COMMAS>
        WORD1,WORD2
        (SYNONYMOUS WORDS3 SEPARATED BY COMMAS)
        (WORD4/s,WORD5/ENDING1/ENDING2,WORD6//ENDING1)
        WORD7/NUMBER1,WORD8//NUMBER2
```

NOTE: Up to 7 characters are permitted in the name of the vocabulary. When sets of endings are used repeatedly, -endings- plus -vocab(s)- may be more convenient than -vocab(s)- with actual endings included.

Modification of judging copy of response

bump — removes the specified characters from the judging copy of the response before judging

```
bump    CHARACTERS   (maximum of 8 characters; use additional
                        -bump- commands for more than 8 characters)
```

put — replaces a character string in the judging copy of the response with another character string

```
put     STRING1=STRING2   (replaces STRING1 with STRING2)
```

putd — similar to -put- but uses the first character in the tag as the separator between strings

```
putd    /STRING1/STRING2/   (separator is /)
putd    ,STRING1,STRING2,   (separator is ,)
```

putv — similar to -put- but works with stored strings

```
putv    ARG1,ARG2,ARG3,ARG4

        ARG1 = starting variable of string (left-justified)
        ARG2 = number of characters in string
        ARG3 = starting variable of replacement string (left-justified)
        ARG4 = number of characters in replacement string
```

NOTE: Maximum number of characters in a string for -put-, -putd-, and -putv- is 5Ø. If the replacement operations cause the judging copy of the response to exceed 3ØØ characters, judging terminates with a "no" judgment.

close — takes characters stored in the right-most six bits from successive variables and makes a judging copy for use with judging commands; often paired with -open-; the end of the character string is indicated by the specified number of characters or by six bits equal to zero (oØØ), whichever is attained first

```
close   STARTING VAR,NUM CHARACTERS   (use n-variables)
```

loada — takes the characters stored in the specified variable(s) by -pack-, -storea-, or -calc- and makes a judging copy; the end of the character string is indicated by the specified number of characters or by six bits equal to zero (oØØ), whichever is attained first

```
loada   STARTING VAR,NUM CHARACTERS (opt)   (number of
           characters, if omitted, is 1Ø)
```

Modification of judging procedure

NOTE: The various -specs- options do not affect all judging commands. Commands affected by each -specs- option are indicated by number from the following list.

judging commands affected by -specs-

1. -match-
2. -answer-, -wrong-, -answerc-, -wrongc-
3. -vocabs-, -concept-, -miscon-
4. -vocab-, -concept-, -miscon-
5. -exact-, -exactc-, -exactv-
6. -ansv-, -wrongv-, -ansu-, -wrongu-, -store-, -storeu-
7. -storen-
8. -storea-

specs allows control over processing of responses; also serves as a marker for execution of subsequent regular commands after judging is complete

specs allwords (treats integers like letters [rather than numbers] so that a number-letter boundary is not like a word-word boundary or punctuation)
(with 1, 2, 3, 4 above)

specs alphxnum (treats a letter-number boundary like a word-word boundary or like punctuation)
(with 1, 2, 3, 4, 7 above)

specs bumpshift (removes shift codes from the judging copy of the response)
(with all commands above)

specs exorder (checks the order of ignorable words)
(with 2 above)

specs holdmark (prevents markup of student response but stores the markup information for later display)
(with all commands above where markup is done: 2, 3, 4)

specs nodiff (turns off the numeric difference judger, which treats a numerical response as a "misspelling" if it is within 10% of the correct response; no spelling markup is done)
(with 2 above)

specs nomark (turns off answer markup)
(with all commands above where markup is done: 2, 3, 4)

```
specs   nookno      (prevents appearance of "ok" and "no")
        (with all commands above)

specs   noops       (prevents use of mathematical operations in a
                    numerical response)
        (with 6 above)

specs   noorder     (turns off the order judger; allows any
                    word order; no order or keyword markup is done)
        (with 2, 3, 4 above)

specs   nospell     (turns off the spelling judger; no spelling
                    markup is done)
        (with 2 above)

specs   novars      (prevents use of variables defined in
                    define set "student")
        (with 6 above)

specs   okassign    (allows assignment of a value to a variable
                    defined in define set "student")
        (with 6 above)

specs   okcap       (allows a capitalized word in the response to
                    match a non-capitalized word in the tag of a
                    response-matching command or in the vocabulary)
        (with 1, 2, 3 above)

specs   okextra     (allows extra words in the response;
                    caution--words not in -vocabs- may be
                    treated as spelling errors)
        (with 2, 3, 4 above)

specs   okspell     (allows any reasonable spelling)
        (with 1, 2, 3 above)

specs   toler       (allows 1% tolerance in a numerical response)
        (with 1, 2 above)

specs   (B)         (acts only as a marker)

specs   nookno,okspell,noorder   (may combine tags)
```

Note: The following system variables are set properly even if use of the -specs- tag causes the response to match the tag of a response-matching command.

-specs- tag	system variable
okspell	spell
okcap	capital
okextra	extra
noorder	order

Storing judging copy of response

NOTE: Commands -store-, -storeu-, and -storen- terminate judging with a "no" judgment if an error is found in the form of the response.

store calculates the numerical value of the response and stores it in the variable specified in the tag

```
store   VAR
```

storeu similar to -store- but also evaluates dimensionality of units

```
storeu  VAR FOR STORING NUMBER,STARTING VAR FOR STORING POWERS
        OF DIMENSIONS   (see -define-: units; use v-variables to
           store the powers of the dimensions; powers are stored
           in the order in which the primary units are defined)
```

storen searches for and evaluates simple numerical expressions (without variables) in the response, which may also contain non-numeric characters; stores numerical results in the specified variables one at a time; removes numerical parts of the response from the answer buffer and replaces them with blanks; numerical parts must be set off from letters by spaces or punctuation; if no numerical expression is found, the variables are set to Ø and judging ends with a "no" judgment; each -storen- increments "anscnt"

```
storen  VAR1
storen  VAR2
  :
  :
```

storea stores characters from the response, left-justified, in the specified variable(s), 1Ø characters per variable; unused character positions are filled with octal zeros

```
storea  STARTING VAR,NUM CHARACTERS (opt)   (number of
           characters, if omitted, is 1Ø)
```

Note: Use n-variable(s) for storing the string if subsequent comparison for equality with another string is done. (If the character string is stored for other purposes, v-variables are acceptable.) Segmented variables cannot be used with -storea-.

open places the characters in the response, one-by-one, in the right-most six bits of successive variables starting at the specified variable

```
open    STARTING VAR   (use n-variables)
```

Matching judging copy of response

NOTE: In this sub-section references to response mean judging copy of the response.

NOTE: Up to 39 "words" (entries separated by space or punctuation) are permitted in responses with -match-, -answer-, -wrong-, -answerc-, -wrongc-, -concept-, and -miscon-. If the number of words exceeds 39, judgment is "no" and "anscnt" is set to -2.

match checks the response against the arguments in the tag and sets a variable to the relative position of the matched character string; if no match is found, the variable is set to -1 and judgment is "no" ("judged" set to +1); otherwise judgment is "ok" ("judged" set to -1); always ends judging

for example:

```
match   VAR,WORDØ,WORD1,PHRASE*WORD2,WORD3
match   VAR,(WORDØ,SYNONYMØ),(WORD1,(LISTNAME1)),((LISTNAME2))
```

answer compares the response with the tag; checks for word order, spelling, capitalization, extra words, and numeric tolerance unless altered by -specs-; sets "judged" to -1 if response matches tag

for example:

```
answer  WORDS AND PHRASE*WORDS   (blank tag matches a response in
           which nothing is entered or which contains only spaces
           and punctuation)
answer  <EXTRA WORDS>(SYNONYMOUS WORDS1 AND PHRASE*WORDS1
        SEPARATED BY COMMAS)(SYNONYMOUS WORDS2 and PHRASE*WORDS2
        SEPARATED BY COMMAS)WORD3
answer  <<LISTNAME1>>((LISTNAME2),WORD2)
answer  RESPONSE1;RESPONSE2;RESPONSE3   (each argument may have any
           of the preceding forms for the tag of -answer-; synonymous
           responses for the same argument are separated by commas)
```

Note: Tag must not contain punctuation (or symbols changed to "punc" in the -change- command), although the student's response may.

wrong compares the response with the tag; performs checks available with -answer-; sets "judged" to Ø if response matches tag

```
wrong   WORDS AND PHRASE*WORDS
```

Note: Options for the tag of -wrong- are the same as for the tag of -answer- but for an incorrect student response.

answerc compares the response with one of several arguments in the tag, depending on the rounded value of the conditional expression; performs the checks available with -answer-; sets "judged" to -1 if response matches the required argument

answerc EXPR;RESPONSEM;RESPONSEØ;;RESPONSE2 (the arguments may have any of the forms allowed in the tag of -answer-; a blank argument indicates no anticipated response for that value of the conditional expression)

wrongc same options as for -answerc- but for an incorrect response; sets "judged" to Ø if response matches the required argument

wrongc EXPR;RESPONSEM;;;RESPONSE2;RESPONSE3

concept compares the response with the tag; -vocab- or -vocabs- provides synonyms; sets "judged" to -1 if response matches tag

concept WORDS AND PHRASE WORDS (no asterisk in phrases; blank tag matches a response which is blank or which contains only ignorable words from the vocabulary)
concept WORD1 WORD2,VAR1⇐WORD1,VAR2⇐WORD2 (detects which synonym is entered if the vocabulary is appropriately set up)

miscon same options as for -concept- but for an incorrect response; sets "judged" to Ø if response matches tag

miscon WORDS AND PHRASE WORDS

exact compares the response with the tag for an exact character string match; sets "judged" to -1 if response matches tag

exact STRING (blank tag matches a response in which nothing is entered)

exactc compares the response for an exact character string match with one of several arguments in the tag depending on the rounded value of a conditional expression; sets "judged" to -1 if response matches the required argument

exactc EXPR,STRINGM,STRINGØ,,STRING2 (blank argument matches a response in which nothing is entered)

exactv compares the response with a stored character string for an exact match; the end of the character string is indicated by the specified number of characters or by six bits equal to zero (oØØ), whichever is attained first; sets "judged" to -1 if response matches tag

exactv STARTING VAR OF STORED STRING,NUM CHARACTERS (opt)
(number of characters, if omitted, is 1Ø; if the number of characters is Ø, the response is judged correct if nothing is entered)

NOTE: With the following four commands (-ansv-, -wrongv-, -ansu-, -wrongu-), tolerance is optional. When tolerance is omitted, the default is $1Ø^{-9}$ if the absolute value of the tag value is less than approximately 1ØØ or $(1Ø^{-11} \times |\text{tag value}|)$ if the absolute value of the tag value is greater than approximately 1ØØ. These commands cannot judge values smaller in absolute value than $1Ø^{-9}$ since any response less than $1Ø^{-9}$ will then match the tag.
Tolerance may be absolute deviation or percent deviation.

ansv checks a numerical response against the first argument in the tag, with tolerance, if given, set by the second argument; sets "judged" to -1 if response matches tag (within the tolerance)

ansv CORRECT VALUE,TOLERANCE

wrongv similar to -ansv- but for the incorrect numerical response; sets "judged" to Ø if response matches tag (within the tolerance)

wrongv INCORRECT VALUE,TOLERANCE

ansu similar to -ansv- but checks for correct units; sets "judged" to -1 if response matches tag (within the tolerance)

ansu NUMBER AND UNITS,TOLERANCE

wrongu similar to -ansu- but for incorrect response; sets "judged" to Ø if response matches tag (within the tolerance)

wrongu NUMBER AND UNITS,TOLERANCE
wrongu NUMBER,TOLERANCE (may be used to indicate that units are missing)

NOTE: For applications of -ansu- and -wrongu- see -storeu- and -define-. Commands -ansv- and -wrongv- accept defined units in the student's response but do not compare with author's units.

NOTE: One touch square is 32 dots on each side (or 4 characters in width and 2 lines in height).

ntouch specifies the location of a rectangle for a touch response; sets "judged" to -1 if the specified rectangle is touched (see -enable- and -disable-)

ntouch AREA1;AREA2;AREA3;... (blank tag matches any touch input)

Note: AREA may be: COARSE,WIDTH IN CHARACTERS,HEIGHT IN LINES
or FINEX,FINEY,WIDTH IN DOTS,HEIGHT IN DOTS
COARSE or FINEX,FINEY is the screen location of the lower left corner of a rectangle of specified width and height. Width and height are optional and assumed to be 1 if omitted.

ntouchw same options as -ntouch- but for an incorrect touch response; sets "judged" to Ø if the specified rectangle is touched

ntouchw AREA1;AREA2;AREA3;AREA4;...

touch specifies the location of the touch square for a touch response; sets "judged" to -1 if the touch-location is matched (within the tolerance); see -enable- and -disable-

touch CENTER COARSE,TOLERANCE IN TOUCH SQUARES (opt)
(tolerance, if omitted, is 1; blank tag matches any touch input)
touch LOWER LEFT COARSE,WIDTH IN TOUCH SQUARES,HEIGHT IN TOUCH SQUARES
touch LOCATION1,TOLERANCE1;LOCATION2,TOLERANCE2;... (up to 2Ø one-, two-, or three-argument touch locations may be specified)

touchw same options as -touch- but for an incorrect touch response; sets "judged" to Ø if the touch-location is matched (within the tolerance)

touchw CENTER COARSE,TOLERANCE (opt)
touchw LOWER LEFT COARSE,WIDTH IN TOUCH SQUARES,HEIGHT IN TOUCH SQUARES

or (no tag) placed on the line between response-matching commands to provide alternative responses for the same value of "anscnt"

ans (no tag) allows use of the ANS key; terminates judging only if ANS is pressed; otherwise normal judging occurs; -ans- must be the first judging command following the -arrow- command

Information on specific words in response

NOTE: In the following commands (-getword-, -getmark-, -getloc-, -compare-), a "word" is an entry set off by spaces or punctuation from surrounding characters. (See -specs allwords-, -specs -alphxnum- and -change symbol- for additional options in specifying word boundaries.)

getword (regular command) allows storage of individual words in a response

getword ARG1,ARG2,ARG3,ARG4 (opt)

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = starting variable for storing the word (up to 10 characters per variable)
ARG3 = variable for storing the actual number of characters in the word (=0 if ARG1>"wcount")
ARG4 = maximum number of characters to be stored in ARG2 (optional; if omitted, value is 10)

Note: Words that are stored are not removed from the judge buffer.

getmark (regular command) used after judging a response to give markup information on individual words in the response

getmark ARG1,ARG2

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = variable containing markup information
= -2 if no markup is done with the response-matching command used
= -1 if the position of the word is out of bounds (i.e., if ARG1>"wcount")
= 0 if there are no errors in the word
> 0 bits in ARG2 are set according to the error(s), starting at the right-most bit (subscript "2" indicates the number is in binary notation):
(1_2) a word preceding this word is missing
(10_2) the word is out of order (too far right)
(100_2) there is a capitalization error
($1\ 000_2$) the spelling is incorrect
($10\ 000_2$) the word is part of a broken phrase
($100\ 000_2$) the word is an extra word
($1\ 000\ 000_2$) this word is the last word, and a word which should follow is missing

getloc (regular command) gives the screen position of the beginning (and end, if requested) of the specified word in the response

getloc ARG1,ARG2,ARG3,ARG4 (opt),ARG5 (opt)

ARG1 = relative position of the word in the response (first word is 1, second word, 2, etc.)
ARG2 = variable for storing the finex screen position of the beginning of the word (= -1 if ARG1 > "wcount")
ARG3 = variable for storing the finey screen position of the beginning of the word
ARG4 = variable for storing the finex screen position of the end of the word (optional)
ARG5 = variable for storing the finey screen position of the end of the word (optional)

compare (regular command) compares two words for spelling

compare AUTHOR WORD,STUDENT WORD,VAR FOR STORING RESULT CODE

result = -1 if words are different
= Ø if words are identical
> Ø if words are misspellings of each other (smaller value indicates a closer match)

Note: The first two arguments may also be variables. The words must be stored in the variables in the same manner, e.g., both words left-justified or both right-justified. If the words are stored with -storea- or -pack-, they will be left-justified. If a word itself is given, rather than the variable(s), it should be enclosed in single quotes for left-justification or double quotes for right-justification.
System variables "spell" and "capital" are set appropriately if result value ≥ Ø:
"spell" is set to Ø if result > Ø;
"capital" is set to Ø if only one word is capitalized

Unconditional judgment

ok (no tag) judges any response "ok"; sets "judged" to -1

no (no tag) judges any response "no"; sets "judged" to +1

ignore (no tag) erases and ignores any response; stops further processing and waits for a new response

Reference to other units which may contain judging commands

join causes execution of the specified unit without change of main unit; commands following -join- are executed; -join- is executed in all states: regular, judging, and search (see also description under SEQUENCING, Automatic sequencing)

```
join    UNIT NAME
join    NAME,VAR← INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR (opt)
join    EXPR,NAMEM,NAMEØ,x,NAME2,q   (example of conditional form)
```

iarrow (regular command) the specified unit is executed after each subsequent -arrow- in a unit just before the first judging command is executed

```
iarrow  UNIT NAME
iarrow  (B) or iarrow q   (clears the -iarrow- setting for
           subsequent -arrow- commands in the unit)
iarrow  EXPR,NAMEM,NAMEØ,q,NAME2,x   (example of conditional form;
           maximum of 1ØØ arguments in the conditional tag)
```

Note: The -iarrow- setting is equivalent to -join- after each subsequent -arrow- (just before the first judging command); the specified unit is executed in all states.

iarrowa (regular command) similar to -iarrow- but is associated with -arrowa-; see -iarrow- for restrictions

```
iarrowa UNIT NAME
```

Alteration of judgment

judge (regular command) alters the judgment rendered by judging commands

judge	ok	(sets judgment to "ok"; sets "judged" to -1; continues processing regular commands)
judge	no	(sets judgment to "no" [unanticipated]; sets "judged" to +1; continues processing regular commands)
judge	wrong	(sets judgment to "no" [anticipated]; sets "judged" to Ø; continues processing regular commands)
judge	okquit	(sets judgment to "ok"; sets "judged" to -1; terminates processing at that arrow except for regular commands following -specs-)
judge	noquit	(sets judgment to "no"; sets "judged" to +1; terminates processing at that arrow except for regular commands following -specs-)
judge	quit	(leaves judgment unchanged and terminates processing at that arrow except for regular commands following -specs-; allows the student to proceed to the next arrow even if judgment on the current arrow is "no")
judge	ignore	(stops all processing, erases the response, and waits for a new response)
judge	exit	(rescinds judgment and waits for additional keys)
judge	continue	(resumes judging using the modified response, as altered by -bump-, -put-, -specs-, -match-, -storen-, etc.; resumes processing judging commands)
judge	rejudge	(resumes judging using the original, unmodified response and clears the -specs- setting; resumes processing judging commands)
judge	EXPR,x,no,ignore,ok	(example of conditional form; argument x leaves judgment unchanged)

Alteration of feedback

okword (regular command) permits "ok" message to be changed

okword NEW WORD FOR USE WITH "OK" JUDGMENT (may be blank)

noword (regular command) permits "no" message to be changed

noword NEW WORD FOR USE WITH "NO" JUDGMENT (may be blank)

NOTE: Tags of -okword- and -noword- may have up to 9 characters.
A space is automatically provided before the message.
Commands -okword- and/or -noword- may be placed anywhere in the lesson. Once they are executed, they are in effect until execution of another -okword- and/or -noword- command.

markup (regular command) (no tag) used with -specs holdmark- to display markup information that was inhibited with -specs holdmark-

System variables for judging

judging in general

anscnt number of response-matching commands encountered at an arrow before the response is matched; also set by -storen-; otherwise,
= -2 if the student's response contains more than 39 words
= -1 if no tag is matched
= Ø for a store error
zeroed for each -arrow- and each -specs- command

ansok = -1 if the response is a satisfactory match to the preceding response-matching command
= Ø otherwise;
in particular, after -no-,
= -1 if there is no match to a previous response-matching command
= Ø if the match is poor

jcount number of internal 6-bit character codes in the response

judged = -1 for any "ok" judgment
= Ø for any "wrong" judgment (anticipated "no")
= +1 for any "no" judgment (unanticipated "no")

key
ztouchx [See descriptions under system variables for sequencing.]
ztouchy

ntries number of attempts on the current arrow

verbal responses

judging commands which affect system variables for verbal responses:

1. -match-
2. -answer-, -wrong-, -answerc-, -wrongc-
3. -vocabs-, -concept-, -miscon-
4. -vocab-, -concept-, -miscon-

capital = -1 if there are no capitalization errors, = Ø otherwise (with 1, 2, 3 above)

entire = -1 if all important words are present, = Ø otherwise (with 2 above)

extra = -1 if there are no extra words in the response, = Ø otherwise
(with 2, 3, 4 above)

order = -1 if word order is ok, = Ø otherwise
(with 2 above)

phrase = -1 if there are no phrase errors, = Ø otherwise
(with 1, 2, 3 above)

spell = -1 if spelling is ok, = Ø otherwise
(with 1, 2, 3 above)

wcount number of words in the response
(with all above)

numerical responses

These system variables are set with -ansv-, -wrongv-, -ansu-, -wrongu-, -store-, -storeu-, -compute-, and the -calc- type commands.

formok gives diagnostics on errors in mathematical expressions

= -1 if the expression is ok
= Ø if there is a bad function argument or index
= 1 if there is an illegal character
= 2 if there are unbalanced parentheses
= 3 if there are too many decimal points
= 4 if there are variables not defined in define set "student"
= 5 if a symbol involving $ is not a logical or a bit operator
= 6 if the expression has bad form
= 7 if a value is assigned to a non-variable
= 8 if an octal constant contains digit 8 or 9
= 9 if there is an error in an alpha string
= 1Ø if a number has too many digits
= 11 if an array index is out of bounds
= 12 if there are variables with -specs novars-
= 13 if there are operations with -specs noops-
= 14 if there are assignments without -specs okassign-
= 15 if units in the response are used incorrectly
= 16 if too much computing is attempted
= 17 if the expression is too deep in nested functions
= 18 if a function has the wrong number of arguments
= 6Ø if too many temporary variables are needed during processing
= 62 if compiled expressions make the unit too long
= 63 if there are too many literal numbers in the expression
= 66 if indexing is too deep in a variable which is assigned a value

opcnt number of arithmetic operations and functions in the response

varcnt number of defined variables and functions (define set "student") in the response

Commands in this section are legal only in a site lesson.

Site commands

These commands, all of which have -site- in the command field, give information on the specified logical site.

site set — specifies the logical site for subsequent -site- commands; a later -site set- command overrides an earlier one

site set,'SITENAME'

Note: zreturn = -1 if -site set- was executed successfully
= Ø if the lesson is not a site lesson for the specified sitename

site info — stores current site ECS information for the site specified by a preceding -site set-

site info,STARTING VAR FOR STORING INFORMATION

Note: Information consists of:

n(x) or nc(x) = starting variable
n(x) contains the base ECS allotment
n(x+1) " the ECS currently allotted
n(x+2) " the ECS currently in use
n(x+3) " the number of active terminals at the site

zreturn = -1 if -site info- was executed successfully
= Ø if no site has been set by -site set-

site active — stores the physical station numbers of the specified number of active stations on the site

site active,STARTING STATION NUMBER,STARTING VAR FOR STORING STATION NUMBERS,NUM ACTIVE STATIONS TO STORE

Note: zreturn = -1 if -site active- was executed successfully
= Ø if no site has been set by -site set-
= +1 if the starting station number is invalid

site stations stores the physical station numbers of the specified number of stations permanently on the site

site stations,STARTING STATION NUMBER,STARTING VAR FOR STORING STATION NUMBERS,NUM STATIONS TO STORE

Note: zreturn = -1 if -site stations- was executed successfully
= Ø if no site has been set by -site set-
= +1 if the starting station number is invalid

Station commands

These commands, all of which have -station- in the command field, give information on individual stations on the specified logical site.

station info stores information on the specified physical station number

station info,STATION NUMBER,STARTING VAR FOR STORING DATA

Note: Information consists of:

n(x) or nc(x) = starting variable
n(x) and n(x+1) contain the sign-on name of the user
n(x+2) contains the user's sign-on course name
n(x+3) " the type of user
n(x+4) " the account name containing the user's course
n(x+5) " session statistics (in 3 2Ø-bit fields--
disk accesses, seconds of CPU, elapsed time)
n(x+6) " the name of the user's lesson or type of activity
n(x+7) " total ECS the lesson is using (in 3 2Ø-bit fields--
storage ECS, common ECS, lesson ECS)
n(x+8) " name of student's router
n(x+9) " ECS used by the router

zreturn = -1 if -station info- was executed successfully
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the station is inactive

station status sets "zreturn" according to the status of the specified physical station number

station status,STATION NUMBER

zreturn = -2 if the station is in the process of signing on
= -1 if the station is active
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the station is inactive
= 4 if a backout of the station is in progress
= 5 if the station is locked out

station send sends the specified message (in -mode rewrite-) to the specified physical station number

station send,STATION NUMBER,SCREEN LOCATION,MESSAGE,NUM CHARACTERS IN MESSAGE

Note: zreturn = -1 if message was sent successfully
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 no message was sent (specified station is inactive or is the station sending the message)

station logout backs out the station (given by the physical station number) (sets "backout" to -2)

station logout,STATION NUMBER

Note: zreturn = -1 if the backout was successful
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the specified station cannot be backed out

station stop1 presses STOP1 at the specified station (sets "backout" to +1 until the station enters another instructional lesson)

station stop1,STATION NUMBER

Note: zreturn = -1 if STOP1 was pressed at the station
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if STOP1 cannot be pressed at the specified station

station off turns off the specified station and prevents further use of the terminal (sets "backout" to -2)

station off,STATION NUMBER

Note: zreturn = -1 if the station was turned off successfully
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 if the specified station cannot be turned off

station on turns on the specified station

station on,STATION NUMBER

Note: zreturn = -1 if the station was turned on successfully
= Ø if no site has been set by -site set-
= 1 if the starting station number is invalid
= 2 if the specified station is not on the site
= 3 the station was already active

Basic display

at specifies starting position of display on the panel

```
at      COARSE
at      FINEX,FINEY
```

Note: The following formulas convert between character grid and fine grid.

```
finex = 800frac(coarse/100) - 8
finey = 512 - 16int(coarse/100)
coarse = 100{1 + int[(511 - finey)/16]} + int(finex/8) + 1
```

atnm like -at- but does not reset the left margin

```
atnm    COARSE
atnm    FINEX,FINEY
```

write displays text on the panel

```
write   ANY MESSAGE, WHICH MAY CONSIST OF SEVERAL LINES
        AND INCLUDE EMBEDDED INFORMATION.
```

writec displays one of several messages depending on the rounded value of the conditional expression

```
writec  EXPR↕MESSAGEM↕MESSAGE0↕MESSAGE1↕↕MESSAGE3
```

NOTE: The embed feature may be used with -write- and -writec-. See descriptions of the individual commands in this section for information on FORMAT, MINIMUM, and ASTERISK, which are optional.

```
embedded -show-    ⟨ show,EXPR,FORMAT,MINIMUM⟩  or  ⟨ s,EXPR,FORMAT,MINIMUM⟩
embedded -showz-   ⟨ showz,EXPR,FORMAT⟩  or  ⟨ z,EXPR,FORMAT⟩
embedded -showt-   ⟨ showt,EXPR,FORMAT⟩  or  ⟨ t,EXPR,FORMAT⟩
embedded -showe-   ⟨ showe,EXPR,FORMAT,ASTERISK⟩  or  ⟨ e,EXPR,FORMAT,ASTERISK⟩
embedded -showo-   ⟨ showo,EXPR,FORMAT⟩  or  ⟨ o,EXPR,FORMAT⟩
embedded -showa-   ⟨ showa,EXPR,FORMAT⟩  or  ⟨ a,EXPR,FORMAT⟩
embedded -at-      ⟨ at,COARSE⟩ ; ⟨ at,FINEX,FINEY⟩
embedded -atnm-    ⟨ atnm,COARSE⟩ ; ⟨ atnm,FINEX,FINEY⟩
embedded -size-    ⟨ size,EXPR GIVING SIZE OF WRITING⟩
                   ⟨ size,SIZE IN X DIRECTION,SIZE IN Y DIRECTION⟩
embedded -rotate-  ⟨ rotate,EXPR GIVING ANGLE FOR WRITING⟩
                   ⟨ m,w⟩  (write mode)
embedded -mode-    ⟨ m,e⟩  (erase mode)
                   ⟨ m,r⟩  (rewrite mode)
```

NOTE: In the following six commands (-show-, -showz-, -showt-, -showe-, -showo-, -showa-), the general form is

showz EXPR,FORMAT

where FORMAT, which may be an expression, is optional. If FORMAT equals Ø, nothing is displayed.

For displaying arrays the general form is

showt ARRAYNAME,FORMAT (maximum of 16 rows may be displayed)

show displays the value of a variable or an expression with the specified number of significant digits but with suppression of trailing zeros after the decimal point; exponential format is displayed if the number of digits preceding the decimal point exceeds FORMAT by more than 4, or if the absolute value is less than 10^{-4}; MINIMUM is between Ø and 1 and specifies the smallest non-zero value to be displayed (Ø is displayed if the absolute value of the expression is less than MINIMUM)

show EXPR,NUM DIGITS,MINIMUM (FORMAT, if omitted, is 4; MINIMUM, if omitted, is 10^{-9})

showz similar to -show- but displays all digits, including trailing zeros

showz EXPR,NUM DIGITS (FORMAT, if omitted, is 4)

showt displays the value of a variable or an expression in the specified format

showt EXPR,NUM DIGITS PRECEDING DECIMAL POINT,NUM DIGITS FOLLOWING DECIMAL POINT (may be omitted if zero) (FORMAT, if omitted, is 4,3 for v-variable, 8 for n-variable; if the number of decimal places is less than 10, FORMAT may also be expressed as a single decimal number: e.g., 4.3 is equivalent to 4,3)

showe displays the value of a variable or an expression in exponential format with the specified number of significant digits, including a leading blank or a minus sign; an optional third argument specifies the format for the exponent

showe EXPR,NUM DIGITS,ASTERISK (FORMAT, if omitted, is 4; ASTERISK is omitted or =Ø for exponent expressed by superscript, ≠Ø for exponent expressed by 2 asterisks and multiplication sign replaced by one asterisk)

showo displays the value of a variable or an expression in octal notation

showo EXPR,NUM DIGITS DISPLAYED (FORMAT, if omitted, is 21; in embedded -showo- default format is 2Ø)

showa displays characters in the specified variable(s) or specified string

showa STARTING VAR,NUM CHARACTERS (FORMAT, if omitted, is 1Ø)
showa 'STRING' (STRING may contain up to 1Ø characters)

hidden displays hidden as well as visible characters; (special symbols are used to display hidden characters); number of characters includes all 6-bit character codes

hidden STARTING VAR,NUM CHARACTERS (opt) (NUM CHARACTERS, if omitted, is 1Ø)

text displays contents of an alphanumeric buffer, line by line; the end of a line must be indicated by a variable ending with 2 zero codes (i.e., 12 bits equal to Ø: oØØØØ) (embedded zero codes (oØØ) are ignored); not affected by -size- or -rotate-

text STARTING VAR,NUM VARS

erase erases the screen, selectively or entirely

erase abort (causes a full-screen erase and aborts output)
erase (B) or erase NEGATIVE NUMBER (causes full-screen erase but does not abort output)
erase NUM CHARACTERS TO BE ERASED
erase NUM CHARACTERS PER LINE,NUM LINES (causes block erase)

Note: Selective erase is affected by preceding -size-, -gorigin- (and -scalex-, -scaley-), and -rorigin-.

mode specifies terminal writing mode (see also system variable "mode")

mode write (normal writing state; writes only selected dots)
mode erase (erases only selected dots)
mode rewrite (erases and rewrites in one step; does not work with "size" ≠ Ø)
mode EXPR,erase,write,x,erase (example of conditional form; argument x leaves writing mode unchanged)

Note: The mode is reset to "write" after any full-screen erase, in particular at a main unit not containing -inhibit erase-.

color displays screen information in the specified color

```
color   orange   (same as -mode  write-)
color   black    (same as -mode  erase-)
```

size specifies the size of line-drawn characters; remains in effect across main unit boundaries until turned off explicitly (see also system variables "size", "sizex", "sizey")

```
size    EXPR GIVING SIZE OF CHARACTERS
size    SIZE IN X DIRECTION,SIZE IN Y DIRECTION   (sets independent
           sizes in x and y directions)
size    Ø   or   size    (B)   (restores standard writing)
```

Note: Negative "sizex" gives backwards characters and writing; negative "sizey" gives upside down characters and writing. Negative "size" behaves like simultaneous negative "sizex" and negative "sizey".

rotate causes line-drawn characters to be written at the angle specified in the tag; remains in effect across main unit boundaries until turned off explicitly (must be used with -size- tag ≠ Ø)

```
rotate  EXPR GIVING ANGLE IN DEGREES (omit degree symbol;
        measured counter-clockwise from horizontal)
rotate  Ø   or   rotate  (B)   (restores horizontal writing)
```

inhibit temporarily disables certain normal actions of TUTOR in a unit; all settings are cleared at each main unit

```
inhibit anserase     (prevents automatic erasure of answer-
                      contingent message when a response is erased)
inhibit arrow        (prevents plotting of the response arrow)
inhibit blanks       (prevents judging if NEXT is pressed before
                      any characters are typed)
inhibit charclear    (prevents clearing of the charset flag)
inhibit dropset      (prevents the dataset from being released
                      during a jumpout)
inhibit dropstor     (prevents storage from being dropped during
                      a jumpout)
inhibit edit         (prevents use of the EDIT key)
inhibit erase        (prevents normal full-screen erase when
                      proceeding to the next main unit)
inhibit from         (prevents return to the lesson containing
                      this statement via -jumpout  return- or
                      -jumpout  return,return-)
inhibit jumpchk      (prevents ECS check before attempting a
                      jumpout)
inhibit term         (prevents use of the TERM key)
inhibit (B)  or  inhibit clear   (removes effect of previous
                      -inhibit- commands in that main unit)
inhibit clear,arrow,blanks   (may combine tags)
```

delay permits short delays during output

delay DURATION OF OUTPUT DELAY IN FRACTIONS OF A SECOND, UP TO 1 SECOND (accurate to 1/60 second)

Note: This command causes "do-nothing" output to be sent to the terminal for the specified delay time.

char permits specification of specially designed characters for display

char NAME,ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7,ARG8

Note: The character name (NAME) may be a number from 0 to 126 (excluding 63) or a defined name. Arguments ARG1 through ARG8 are numbers which specify which of the 16 dots are lit in each of the 8 columns of the character space.

for example:

```
define  chi=88    $$ load chi on X
char    chi,o4020,o10040,o63000,o16000,o31400,o4020,o10040,0
```

plot displays a special character previously specified by a -char- command

```
plot    NAME
plot    EXPR   (EXPR may have value from 0 to 126, excluding 63)
```

Note: Special characters may also be displayed by pressing the FONT key and then the key(s) where the character(s) are loaded into the terminal memory. Built-in characters are displayed after FONT is pressed again.

Graphics

NOTE: In the following five commands (-dot-, -draw-, -box-, -vector-, -window-), LOCATION is the screen location and may be COARSE or FINEX,FINEY. Coarse grid and fine grid coordinates may be mixed in tags with more than one argument.

dot — draws a dot at the specified screen location

```
dot     LOCATION
```

draw — draws a dot, line, or line-drawn figure; after execution "wherex" and "wherey" are set to the last point plotted

```
draw    LOCATION   (draws a dot; -dot- is faster if many dots are
                    plotted; -draw- is faster if lines are also being drawn)
draw    LOCATION1;LOCATION2   (draws a line)
draw    LOCATION1;LOCATION2;LOCATION3;...   (draws connected lines)
draw    ;LOCATION   (draws a continued line)
draw    LOCATION1;LOCATION2;skip;LOCATION3;LOCATION4
                    ("skip" moves to a new position without plotting)
```

Note: Maximum number of numbers in the tag is 63 ("skip" counts as a number).

box — draws a rectangle with the specified corner locations and thickness

```
box     CORNER LOCATION;OPPOSITE CORNER LOCATION;DOTS THICK(opt)
box     ;CORNER LOCATION;DOTS THICK (opt)   (opposite corner at
                    "wherex", "wherey")
box     CORNER LOCATION   (opposite corner at Ø,Ø; cannot specify
                    thickness with this form of tag)
box     (B)   (equivalent to  -box  Ø,Ø;511,511-)
```

Note: Thickness, if omitted, Ø, 1, or -1, is 1 dot. Negative thickness extends inward; positive thickness extends outward.

vector — draws a vector symbol with specified tail and head locations and head size

```
vector  TAIL LOCATION;HEAD LOCATION;SIZE (opt)
vector  ;HEAD LOCATION;SIZE (opt)  (tail at "wherex", "wherey")
vector  HEAD LOCATION   (tail at Ø,Ø ; cannot specify head size)
vector  Ø,Ø;HEAD LOCATION;SIZE   (tail at Ø,Ø ; head size specified)
```

Note: Size, if omitted, is 1Ø.5 (in dots) for moderate-length vectors. Negative size indicates open arrowhead. Size ≥1 is absolute (in screen dots); size <1 is relative to the length of the vector.

window establishes a rectangular "window" on the screen outside of which line-drawn display is not plotted; remains in effect across main unit boundaries until turned off explicitly

```
window  CORNER LOCATION;OPPOSITE CORNER LOCATION
window  ;CORNER LOCATION  (opposite corner at current "wherex",
           "wherey")
window  CORNER LOCATION   (opposite corner at Ø,Ø)
window  (B)   (clears previous -window- setting)
```

circle draws a circle with the specified parameters; the center is at the current "wherex", "wherey"; after execution "wherex", "wherey" are set to the center for a one-argument tag and to the end of the last line drawn for the three-argument tag

```
circle  RADIUS IN DOTS,START ANGLE,END ANGLE
           (second and third arguments are optional: if omitted,
           START ANGLE is Ø° and END ANGLE is 36Ø°; angles
           are measured in degrees counter-clockwise from
           START ANGLE; degree sign is omitted)
```

circleb same options as -circle- but draws a broken circle

```
circleb RADIUS IN DOTS,START ANGLE,END ANGLE
```

Relocatable graphics

rorigin establishes a "relocatable" origin for use with -rdraw-, -rat-, -rbox-, -rvector-, and -rcircle-

```
rorigin COARSE
rorigin FINEX,FINEY
rorigin (B)   (sets relocatable origin to "wherex", "wherey")
```

Note: Upon entering a lesson, the relocatable origin is automatically set to -rorigin Ø,Ø-.

NOTE: All subsequent relocatable commands are affected by preceding -rorigin-, -size-, and -rotate-.

rat similar to -at- but relative to the -rorigin- location; affected by preceding -size- and -rotate-

```
rat     X-LOCATION,Y-LOCATION
rat     (B)   (equivalent to -rat  Ø,Ø-, i.e., the current
              -rorigin- location)
```

ratnm similar to -rat- but does not reset the left margin (see -atnm-)

```
ratnm   X-LOCATION,Y-LOCATION
```

rdot draws a dot at the specified position relative to the -rorigin- location; position is affected by preceding -size- and -rotate-

```
rdot    X-LOCATION,Y-LOCATION
```

rdraw similar to -draw- but figure is affected by preceding -size- and -rotate- or by -rotate- alone; the last point plotted serves as the location for the next screen activity

```
rdraw   TAG LIKE -draw- EXCEPT WITH RESPECT TO -rorigin- LOCATION
        (i.e., in screen dots from the -rorigin- location)
```

rbox similar to -box- but draws a rectangle relative to the -rorigin- location; affected by preceding -size- and -rotate- (see -box-)

```
rbox    CORNER X,CORNER Y;OPPOSITE CORNER X,OPPOSITE CORNER Y;
        DOTS THICK (opt)
rbox    ;CORNER X,CORNER Y;DOTS THICK (opt)   (opposite corner
          at "wherex", "wherey")
rbox    CORNER X,CORNER Y   (opposite corner at -rorigin- location;
          cannot specify thickness with this form of tag)
```

rvector similar to -vector- but draws vector symbol relative to the -rorigin- location; affected by preceding -size- and -rotate-

```
rvector XTAIL,YTAIL;XHEAD,YHEAD;SIZE (opt)
rvector ;XTAIL,XHEAD;SIZE (opt)   (tail at "wherex", "wherey")
rvector XHEAD,YHEAD   (tail at -rorigin- location)
rvector Ø,Ø;XHEAD,YHEAD;SIZE   (tail at -rorigin- location)
```

rcircle same options as -circle- but is affected by preceding -rotate- and -size-; gives an ellipse if preceded by two-argument -size- with unequal arguments (see -circle-)

```
rcircle RADIUS IN DOTS,START ANGLE,END ANGLE
```

Drawing graphs

gorigin specifies location of the origin of the graph; all other display with graphing commands is relative to this origin

```
gorigin COARSE
gorigin FINEX,FINEY
gorigin (B)   (sets graph origin to "wherex", "wherey")
```

Note: Upon entering a lesson, the origin is automatically set to -gorigin Ø,Ø-.

NOTE: With -axes- and -bounds-, x and y values are in dots relative to the -gorigin- location.

axes specifies lengths of the axes and draws the axes

```
axes    NEGATIVE X,NEGATIVE Y,POSITIVE X,POSITIVE Y
axes    POSITIVE X,POSITIVE Y
axes    (B)   (draws axes specified by the last -axes- or -bounds-)
```

Note: To draw one-quadrant axes (other than both positive axes) with labeling on the outside of the axes, use the four-argument form of the tag with arguments corresponding to missing axes set to Ø.

bounds specifies lengths of the axes but does not draw the axes (i.e., the axes are invisible)

```
bounds  NEGATIVE X,NEGATIVE Y,POSITIVE X,POSITIVE Y
bounds  POSITIVE X,POSITIVE Y
bounds  (B)   (sets up bounds specified by the last -axes- or -bounds-)
```

Note: Upon entering a lesson the boundaries are automatically set to -bounds 511,511-.

NOTE: If any of the next four commands (-scalex-, -scaley-, -lscalex-, -lscaley-) are omitted, a linear scale with length set by the preceding -axes- is assumed.

scalex specifies the maximum value and the value at the origin on the x axis; remains in effect across main unit boundaries until reset

```
scalex  MAXIMUM VALUE OF X,VALUE OF X AT ORIGIN (opt)
            (value at origin, if omitted, is Ø)
```

scaley same options as -scalex- but for the y axis

scaley MAXIMUM VALUE OF Y,VALUE OF Y AT ORIGIN (opt)
(value at origin, if omitted, is Ø)

lscalex specifies the maximum value and the value at the origin on the x axis; the scale between these points is proportional to the logarithm of maximum x divided by the value at the origin; remains in effect across main unit boundaries until reset

lscalex MAXIMUM VALUE OF X,VALUE OF X AT ORIGIN (opt)
(value at origin, if omitted, is 1, i.e., 1ØØ)

lscaley same options as -lscalex- but for the y axis

lscaley MAXIMUM VALUE OF Y,VALUE OF Y AT ORIGIN (opt)
(value at origin, if omitted, is 1, i.e., 1ØØ)

NOTE: Subsequent graphing commands are in appropriate scaled units.

NOTE: For the following four labeling commands,
MARKSIZE = Ø or omitted for normal label marks
MARKSIZE = 1 for major marks extending to bounds of the graph
MARKSIZE = 2 for all marks extending to bounds of the graph
MINOR MARKS may be omitted. If MAJOR MARKS is omitted or set to Ø, the computer chooses the "best" interval.
FORMAT gives the format for the labels and has the same form as that for -showt-, e.g., 1.2. FORMAT is optional; if omitted, the label format is selected automatically.

For labeling log scales:
MAJOR MARKS must be Ø (major marks are automatically plotted every decade)
MINOR MARKS < Ø, minor marks are not plotted
MINOR MARKS = Ø or 3 (or omitted), minor marks are placed at values of 1, 2, 5 within the decade
MINOR MARKS = 5, minor marks are placed at 1, 2, 3, 5, 7
MINOR MARKS = 1Ø, minor marks are placed at 1, 2, 3, 4, 5, 6, 7, 8, 9

labelx draws tick marks and labels the x axis

labelx MAJOR MARKS,MINOR MARKS,MARKSIZE,FORMAT

labely draws tick marks and labels the y axis

labely MAJOR MARKS,MINOR MARKS,MARKSIZE,FORMAT

markx draws tick marks on the x axis with no labels

```
markx   MAJOR MARKS,MINOR MARKS,MARKSIZE,FORMAT
```

marky draws tick marks on the y axis with no labels

```
marky   MAJOR MARKS,MINOR MARKS,MARKSIZE,FORMAT
```

polar causes tags of graphing commands to be interpreted as polar coordinates containing scaled radius and polar angle; may set scales on x and y axes; remains in effect across main unit boundaries until turned off explicitly; polar conversion and scaling must be turned off independently

```
polar   (B)   (turns on polar conversion)
polar   MAXIMUM VALUE OF X AND Y   (turns on polar conversion
           and scales both axes)
polar   MAXIMUM VALUE OF X,MAXIMUM VALUE OF Y   (turns on
           polar conversion and scales axes independently)
polar   NEGATIVE VALUE   (turns off polar conversion but not scale)
```

NOTE: When the tag of subsequent commands is interpreted in polar coordinates, the degree sign must be used if the angle is in degrees. Without the degree sign, the angle is interpreted in radians.

gat similar to -at- but specifies the screen location relative to the -gorigin- location and in scaled units

```
gat     X-LOCATION,Y-LOCATION
gat     DISTANCE,ANGLE   (with -polar-)
gat     (B)   (equivalent to -gat  Ø,Ø-, i.e., the current
              -gorigin- location)
```

gatnm similar to -gat- but does not reset the left margin (see -atnm-)

```
gatnm   X-LOCATION,Y-LOCATION
gatnm   DISTANCE,ANGLE   (with -polar-)
```

gdot draws a dot at the specified position relative to the -gorigin- location and in scaled units

```
gdot    X-LOCATION,Y-LOCATION
gdot    DISTANCE,ANGLE   (with -polar-)
```

graph places a dot or character string centered at the position indicated relative to the -gorigin- location and in scaled units

```
graph   X-LOCATION,Y-LOCATION,STRING (opt)   (maximum of 9
           characters in string; if string is not specified, a
           dot is plotted)
graph   DISTANCE,ANGLE,STRING   (with -polar-)
graph   X-LOCATION,Y-LOCATION;VAR,NUM CHARACTERS (opt)
           (if number of characters is omitted, 10 characters
           are plotted)
graph   DISTANCE,ANGLE;VAR,NUM CHARACTERS   (with -polar-)
```

gdraw like -draw- but relative to the -gorigin- location and in scaled units; after execution "wherex", "wherey" are set to the last point plotted

for example:

```
gdraw   X1,Y1;X2,Y2   (draws a line on the graph)
gdraw   DISTANCE1,ANGLE1;DISTANCE2,ANGLE2   (with -polar-)
```

gbox same options as -box- but draws a rectangle relative to the -gorigin- location; affected by preceding -scalex- and -scaley-

```
gbox    CORNER X,CORNER Y;OPPOSITE CORNER X,OPPOSITE CORNER Y;
        DOTS THICK (opt)
gbox    DISTANCE CORNER,ANGLE CORNER;DISTANCE OPPOSITE CORNER,
        ANGLE OPPOSITE CORNER;DOTS THICK   (with -polar-)
gbox    ;CORNER X,CORNER Y;DOTS THICK (opt)   (draws a box
           with opposite corner at current "wherex", "wherey")
gbox    ;DISTANCE CORNER,ANGLE CORNER;DOTS THICK   (with -polar-)
gbox    CORNER X,CORNER Y   (draws a box with opposite corner at
           -gorigin- location; cannot specify thickness with this
           form of tag)
gbox    DISTANCE CORNER,ANGLE CORNER   (with -polar-)
gbox    (B)   (draws a box set by a previous -axes-/-bounds- and
              -scalex-/-scaley-)
```

gcircle same options as -circle- but draws a circle (or ellipse) centered at "wherex", "wherey" relative to the -gorigin- location and in scaled units; draws an ellipse if the -scalex- and -scaley- settings are different (see -circle-)

```
gcircle RADIUS IN DOTS,START ANGLE,END ANGLE
```

gvector same options as -vector- except draws vector symbol relative to the -gorigin- location and in scaled units

```
gvector XTAIL,YTAIL;XHEAD,YHEAD;SIZE (opt)
gvector DISTANCETAIL,ANGLETAIL;DISTANCEHEAD,ANGLEHEAD;SIZE(opt)
          (with -polar-)
gvector ;XHEAD,YHEAD;SIZE (opt)   (tail at "wherex", "wherey")
gvector ;DISTANCE HEAD,ANGLE HEAD;SIZE (opt)   (with -polar-)
gvector XHEAD,YHEAD   (tail at -gorigin- location)
gvector Ø,Ø;XHEAD,YHEAD;SIZE   (tail at -gorigin- location)
gvector LENGTH,ANGLE   (tail at -gorigin- location; with -polar-)
```

Note: Because of the default conditions of -gorigin Ø,Ø- and -bounds 511,511-, -gvector- used without preceding -gorigin- and -bounds- gives the same result as -vector- with fine-grid coordinates.

vbar draws a vertical bar at the specified location relative to the -gorigin- location and in scaled units

```
vbar   X-LOCATION,HEIGHT,STRING (opt)
vbar   DISTANCE BAR TOP,ANGLE BAR TOP,STRING   (with -polar-)
vbar   X-LOCATION,HEIGHT;VAR,NUM CHARACTERS (opt)
vbar   DISTANCE BAR TOP,ANGLE BAR TOP;VAR,NUM CHARACTERS
          (with -polar-)
```

hbar draws a horizontal bar at the specified location relative to the -gorigin- location and in scaled units

```
hbar   LENGTH,Y-LOCATION,STRING (opt)
hbar   DISTANCE BAR END,ANGLE BAR END,STRING   (with -polar-)
hbar   LENGTH,Y-LOCATION;VAR,NUM CHARACTERS (opt)
hbar   DISTANCE BAR END,ANGLE BAR END;VAR,NUM CHARACTERS
          (with -polar-)
```

NOTE: With -vbar- and -hbar-, STRING may have up to 9 characters. If STRING is omitted, a rectangle is drawn. If the character string is stored in a variable and number of characters is omitted, 1Ø characters are drawn.

delta specifies stepsize for subsequent -funct- commands

```
delta   STEPSIZE
```

Note: If -delta- is omitted, the stepsize is set to 1.

funct plots the curve specified in the tag, with the stepsize given by a preceding -delta- or by the stepsize given in the tag

funct FUNCTION EXPR,INDEPENDENT VAR

Note: Range of independent variable is set by boundaries of -axes- (or -bounds-) and -scalex- commands.

funct FUNCTION EXPR,INDEPENDENT VAR⇐ INITIAL,FINAL,STEPSIZE

Note: If initial or final values of the independent variable are beyond previously set boundaries, the latter are used. For polar functions if initial or final value is omitted, it is assumed to be $\emptyset$ or 2π, respectively.

With either form of -funct-, a v-variable is recommended for the independent variable.

NOTE: With -delta- and -funct- select a stepsize that gives a smooth graph but plots quickly. A reasonable lower limit to the stepsize for a graph with linear x axis is:

$|\text{STEPSIZE}| \geq .\emptyset 2 \times |\text{FINAL VALUE} - \text{INITIAL VALUE}|$.

Non-screen

slide operates the slide projector and selects the specified slide

```
slide   SLIDE NUMBER   (value from Ø to 255)
slide   ROW+16COLUMN   (ROW, COLUMN from Ø to 15)
slide   512   (turns off lamp)
slide   256   (closes shutter)
slide   512+SLIDE NUMBER   (selects slide with lamp off)
slide   256+SLIDE NUMBER   (selects slide with shutter closed)
slide   noslide   (selects slide Ø, turns off lamp, closes shutter)
```

audio sends the value of the tag (truncated to 15 bits) to the external device connected to the "audio" jack

```
audio   EXPR
```

play plays the audio device recording at the message location specified

```
play    TRACK,SECTOR,NUM SECTORS   (128 tracks, 32 sectors each)
```

record records a message at the location specified

```
record  TRACK,SECTOR,NUM SECTORS
```

enable (no tag) allows inputs from an external device other than the keyset (e.g., touch panel); must be reset for each -arrow- command in a unit; if used in a unit with no -arrow- commands, allows a touch anywhere on the screen to have same effect as pressing NEXT

disable (no tag) refuses inputs from any external device except the keyset; this is the normal state and is reset for each new -arrow- command or full-screen erase

ext sends the value of the tag (truncated to 15 bits) to an external accessory (or to the accessory at another station if "ext" option has been turned on by the receiving user)

```
ext     EXPR
ext     EXPR,STATION   ("zreturn" is set to -1 if successful,
                        to Ø if not)
```

extout sends the value of the right-most 16 bits of the specified variable(s) to an external accessory; the 16th bit from the right determines how the information is interpreted: 1 for ext, Ø for audio

```
extout  STARTING VAR,NUM VARS   (NUM VARS, if omitted, is 1)
```

saylang specifies the language to be spoken by a phonemic synthesizer which is operated by the terminal (currently only WES [World English Spelling], ipa [International Phonetic Alphabet], and Esperanto are available); currently works only with Votrax model VS-6

saylang LANGUAGE
saylang (B) (turns off subsequent -say- commands)

say specifies the sentence to be spoken by the synthesizer

say SENTENCE OR PHRASE (may include embedded information)

sayc specifies the sentence to be spoken by the synthesizer depending on the value of a conditional expression

sayc EXPR↕PHRASEM↕PHRASEØ↕PHRASE1 ...

Special display

tabset sets tabs which can be used by a student pressing the TAB key

```
tabset  OCTAL NUMBER CONTAINING 1Ø PACKED TAB SETTINGS FROM
        LEFT TO RICHT  (each setting is a 6-bit octal number giving
           the horizontal character position; unused settings to
           the right must be filled with octal zeros)

for example, to set tabs at horizontal character positions
8, 21, 3Ø, 48, 56, and 63, use:

tabset  o1Ø 25 36 6Ø 7Ø 77 ØØ ØØ ØØ ØØ
```

micro specifies microtable definitions used when the student presses the MICRO key and then another key; unless specified, microtable and -micro- command are in the same lesson

```
micro   (Ø),MICROTABLE NAME
micro   (zlesson),MICROTABLE NAME
micro   ,MICROTABLE NAME
micro   MICROTABLE NAME
micro   LESSON NAME,MICROTABLE NAME  (LESSON NAME contains the
           microtable; enclose variable arguments in parentheses)
micro   (B)   (cancels microtable in effect and restores built-in
              microtable definitions)

Note:  zreturn = -1  if the microtable is available
               =  Ø  if the microtable is not found
```

charset causes the specified character set to be loaded into the terminal memory (see -inhibit charclear-); 1-block charset may contain up to 79 characters, 2-block charset up to 126 characters; unless specified, the charset and -charset- command are in the same lesson

```
charset (Ø),CHARSET NAME
charset (zlesson),CHARSET NAME
charset ,CHARSET NAME
charset CHARSET NAME
charset LESSON NAME,CHARSET NAME  (LESSON NAME contains the charset
           blocks; enclose variable arguments in parentheses)
charset (B)   (clears charset flag)

Note:  zreturn = -1  if the character set is loaded successfully
               =  Ø  if the character set is not found
               =  1  if the STOP key is pressed during loading
               =  2  if there is an error in loading
               =  3  if the character set is empty
               =  5  if the variable containing the charset name
                     equals Ø
```

chartst allows check for presence of a character set in the terminal memory; sets "zreturn" to -1 if the charset flag is set and to Ø if not

```
chartst (Ø),CHARSET NAME
chartst (zlesson),CHARSET NAME
chartst ,CHARSET NAME
chartst CHARSET NAME
chartst LESSON NAME,CHARSET NAME  (LESSON NAME contains the
          charset; enclose variable arguments in parentheses)
```

lineset allows use of line-drawn characters, which are affected by preceding -size- and -rotate-; "size" must not be Ø; linechars are accessed by the FONT key or by -altfont on- ; a lineset may be up to 3 blocks long; 1 block may contain up to 128 small linechars; unless specified, lineset blocks and -lineset- command are in the same lesson

```
lineset (Ø),LINESET NAME
lineset (zlesson),LINESET NAME
lineset ,LINESET NAME
lineset LINESET NAME
lineset LESSON NAME,LINESET NAME   (LESSON NAME contains the
          lineset; enclose variable arguments in parentheses)
lineset (B)   (cancels lineset in effect and restores standard
              sized writing)
```

Note: zreturn = -1 if the lineset is attached successfully
= Ø if the lineset is not found
= +1 if there is an error in the lineset

altfont changes font mode of the terminal; affects charsets and linesets

```
altfont on   or   altfont 1   or   altfont alt   (switches terminal
          to alternate font)
altfont off   or   altfont Ø   or   altfont normal  (switches
          terminal to normal font, which is the default state)
```

Note: Tag may be calculated, but it must be exactly Ø or 1. Altfont setting remains in effect across unit boundaries until reset by another -altfont- command or until -jumpout- is executed.

System variables for presenting

mode	= -1 with -mode erase- or -color black- = Ø with -mode rewrite- = +1 with -mode write- or -color orange-	see also commands -mode- and -color-
size	current value of the tag of the single-argument -size- command (see also command -size-)	
sizex	current value of the "x" argument in the two-argument -size- command	
sizey	current value of the "y" argument in the two-argument -size- command	
where	character-grid location for next display	
wherex	fine-grid x location for next display	
wherey	fine-grid y location for next display	

Router lesson

route used in a router lesson to specify to which unit in the router lesson the student is sent upon leaving the instructional lesson

```
route   end lesson,UNIT NAME   (exit via -end  lesson- or -jumpout  q-)
route   error,UNIT NAME     (exit via execution error)
route   finish,UNIT NAME    (exit via STOP1)
route   resignon,UNIT NAME (opt)   (upon STOP1 exit from
           a lesson, provides the student with a choice page
           offering the option to sign off completely or to
           continue working [i.e., return to the router, to the
           specified unit, if given, or to the first unit])
```

routvar (non-executable) sets up special variables in a router lesson which can be used only in the router lesson; they are referenced by vr and nr

```
routvar NUM VARS    (maximum of 5Ø variables)
```

allow used in a router lesson to specify that router common and/or router variables may be referenced in the instructional lesson

```
allow   read     (read-only access to router common)
allow   write    (read and write access to router common)
allow   read rvars   (read-only access to router variables)
allow   (B)    (clears last setting of -allow-)
```

System router

NOTE: The following commands may be used in any context but are particularly useful with the system router.

lesson sets the system variable "ldone" to indicate whether a lesson is considered complete by the system router

```
lesson  completed     (sets "ldone" to -1)
lesson  incomplete    (sets "ldone" to Ø)
lesson  no end    (sets "ldone" to +1; may be used in lessons with
                   no logical end)
lesson  EXPR,completed.incomplete,x,no end    (example of
          conditional form; argument x leaves "ldone" unchanged)
```

score places value of the tag, rounded to the nearest integer, into the system variable "lscore"

```
score   EXPR    (value from Ø to 1ØØ)
score   (B)     (sets "lscore" to -1)
```

status places the value of the tag, rounded to the nearest integer, into the system variable "lstatus"; allows a student to reestablish a status (to some extent) upon returning to a lesson after having entered other lessons

```
status  EXPR
```

System variables for routing

errtype = Ø for unknown error type
= 1 for execution error
= 2 for fatal condense error or for attempted jumpout to a router not specified for the course
= 3 for memory exceeded
= 4 for error in the finish unit of the instructional lesson
= 5 for exit from the condense queue via STOP1

ldone = -1 if the student has encountered -lesson completed- or has pressed NEXT after encountering -end lesson-
= Ø if the student has encountered -lesson incomplete- or has never entered the lesson
= 1 if the student has encountered -lesson no end-
= 2 if the student has started but not finished the lesson (with the system router only)

lscore rounded value of the tag of -score- (value from Ø to 1ØØ); equal to -1 if the student has not previously worked the lesson or if not specifically set by -score- or if set by -score (B)-; initially set to Ø for a student not routed by the system router

lstatus rounded value of the tag of -status-

rcallow = Ø for no access to router common
= 1 for -allow read-
= 2 for -allow write-

router name of the router lesson

rstartl name of the lesson from the last -restart- command

rstartu name of the unit from the last -restart- command

rvallow = Ø for no access to router variables
= 1 for -allow read rvars-

NOTE: The following system variables contain alphabetic information (left-justified) and must be displayed with -showa-: "router", "rstartl", and "rstartu".

Basic sequencing

unit — names and initiates a section of a lesson (called a unit) which may be branched to by other sequencing commands

unit NAME (maximum of 8 characters in NAME; maximum length of a unit is 500 condensed words)

Note: This command may have a form with arguments:

unit NAME(VAR1,VAR2,VAR3,...) (up to 10 arguments)

initial entry unit (ieu) refers to commands preceding the first -unit- command in the lesson; these are always executed whenever and wherever the lesson is entered (except for a router lesson)

entry — names a section of a lesson which may be branched to by other sequencing commands; does not affect the flow of execution of the unit in which the -entry- command is placed except that -entry- may not be placed within the range of -branch-, -doto-, -if-, or -loop-; no keypress is required to execute commands following -entry-; no full-screen erase or other main-unit initializations occur following -entry- when it is executed within a unit

entry NAME (maximum of 8 characters in NAME)

NOTE: A lesson may have up to 395 different units referenced by -unit- and -entry-. No unit may be named "q" or "x".

Automatic sequencing

NOTE: The following commands (-jump-, -goto-, -do-, and -join-) may have a conditional form, e.g.,

```
goto    EXPR,NAMEM,NAMEØ,NAME1,x,NAME3,q
do      EXPR,NAMEM,NAMEØ,x,NAME2,q,VAR← INITIAL,FINAL,STEP
```

Argument x is equivalent to absence of the command; argument q is equivalent to a branch to an empty unit. Special case occurs with -do q- (-join q-), which is equivalent to -goto q-. For iterative -do-, q terminates the -do-, and x indicates no iteration is done for that value of the conditional expression. Argument q is not valid with -jump-. Up to 1ØØ arguments are permitted in the conditional tag.

These commands may pass up to 1Ø arguments, e.g.,

```
goto    NAME(VALUE1,VALUE2,,VALUE4)   (values may be expressions)
:
unit    NAME(VAR1,VAR2,VAR3,VAR4)     (VAR3 is unchanged)
```

jump — causes execution of the unit named in the tag with a full-screen erase (unless the erase is prevented: see -inhibit erase-) and change of main unit; initializations associated with entering a main unit are performed

```
jump    UNIT NAME
```

goto — causes execution of the unit named in the tag without a screen erase, without change of main unit, and without other main-unit initializations; there is no further execution of commands in the original unit except during the judging process

```
goto    UNIT NAME
```

do — (insertion) causes execution of the unit named in the tag without screen erase or change of main unit; returns to the original unit to execute commands following -do-

(iteration) causes repeated execution of unit(s) named in the tag while changing a counter; otherwise same as insertion -do-

```
do      UNIT NAME
do      NAME,VAR← INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR (opt)
           (stepsize, if omitted, is +1; stepsize may be negative;
           loop variable is undefined after completion of the loop)
```

Note: Nested -do- and -join- levels may be up to 1Ø deep.

join similar to -do- but is executed during judging and during search for additional -arrow- commands following an "ok" judgment

```
join    UNIT NAME
join    NAME,VAR← INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR (opt)
```

exit permits termination of -do- or -join- sequences

```
exit    (B) or  exit    NEGATIVE VALUE   (exit from all levels
                                          of -join- and -do-)
exit    EXPR GIVING NUM LEVELS
exit    Ø   (causes no exit)
```

iferror specifies the unit to execute via a -goto- if an error is found in the execution of a subsequent calculation in a unit

```
iferror UNIT NAME
iferror (B)  or  iferror q   (turns off -iferror- setting for
                              remainder of unit)
iferror EXPR,NAMEM,NAMEØ,q,NAME2,x   (example of conditional form;
          maximum of 1ØØ arguments in conditional tag)
```

imain specifies the unit to execute via a -do- at the start of every main unit in the lesson; later occurrence of the command overrides an earlier setting

```
imain   UNIT NAME
imain   (B) or  imain   q   (turns off -imain- setting for
          remainder of lesson or until reset)
imain   EXPR,NAMEM,NAMEØ,q,NAME2,x   (example of conditional form;
          maximum of 1ØØ arguments in conditional tag)
```

NOTE: The following two directives (-branch-, -doto-) are calc-type directives which permit branching or looping within a unit. When the directive is in the command field, it behaves like a -calc- command. In the tag field, the directive is part of a continued -calc-. In both cases non-calculation commands are permitted within the -branch- or -doto- loop.

branch permits branching within a unit (the statement label must start with a number and may contain up to 7 characters)

for example:

```
5a      VAR← EXPR
:
:
branch  EXPR,5a,x,q   (argument x causes fall-tnrough to the next
                       line in the unit; argument q causes
                       execution of the next non-calc command)

calc    VAR← EXPR
6test   VAR← EXPR
        branch  6test
        branch  x    (causes fall-through to next line in the -calc-)
        branch  q    (causes execution of next non-calc command)
        branch  EXPR,q,x,6test,x   (example of conditional form)
```

Note: -branch q- allows a branch to -branch- or -doto- in the command field but not to -branch- or -doto- in the tag field.

doto permits looping within a unit (the statement label must start with a number, may contain up to 7 characters, and must have a blank tag)

for example:

```
doto    2sync,VAR← INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR
:
:
2sync   (B)


calc    VAR← EXPR
        doto  4run,VAR← INITIAL EXPR,FINAL EXPR,STEPSIZE EXPR
                :
                :
4run    (B)
```

Note: Stepsize, if omitted, is +1. Stepsize may be negative. Value of the loop variable is undefined after completion of the loop.

NOTE: The following four commands (-if-, -elseif-, -else-, and -endif-) permit branching within a unit.

if performs a branch based on the rounded value of the tag expression: rounded value < Ø causes fall-through to the next line; rounded value ≥ Ø causes branch to the next -elseif-, -else-, or -endif- at the same level; code following -if- must be indented (up to the next -elseif-, -else-, or -endif- at the same level) and marked with the indent symbol; range of -if- must be terminated by -endif- at the same level

```
if      EXPR
```

elseif provides an alternative branch within the range of the preceding -if- at the same level; subsequent code follows same indenting rules as -if-

```
elseif  EXPR
```

else (no tag) provides a branch if the rounded value of the tag of the preceding -if- or -elseif- at the same level is ≥ Ø; subsequent code follows same indenting rules as -if-

endif (no tag) marks the end of the range of the preceding -if- at the same level

NOTE: Following is an example demonstrating placement of these commands.

```
if      nl<4
.       write   first branch            $$ executed if nl<4
.       calc    n2← 34                  $$ executed if nl<4
elseif  nl=4                            $$ executed if nl≥4
.       write   second branch           $$ executed if nl=4
.       do      someunit                $$ executed if nl=4
else                                    $$ executed if nl>4
.       write   default branch          $$ executed if nl>4
.       if      nl>6                    $$ executed if nl>4
.       .       write   special branch  $$ executed if nl>6
.       endif                           $$ end of range of -if  nl>6-
endif                                   $$ end of range of -if  nl<4-
```

NOTE: The following four commands (-loop-, -endloop-, -outloop-, and -reloop-) permit looping within a unit.

loop initiates a loop based on the rounded value of the tag expression; rounded value < Ø causes execution of subsequent commands (which must be indented and marked with the indent symbol) up to -endloop- at the same level of indentation as -loop-; rounded value ≥ Ø causes execution of the first command after the -endloop- at the same level as -loop-; range of -loop- is marked by -endloop- at the same level

```
loop    EXPR    (blank tag is equivalent to negative value)
```

endloop (no tag) marks the end of a loop initiated by the previous -loop- command at the same level of indentation; causes a branch back to the previous -loop- command at the same level

outloop based on rounded value of the tag expression, causes exit from the range of -loop- at the same level of indentation; rounded value < Ø causes execution of the first command after -endloop- at the same level; rounded value ≥ Ø causes fall-through to following command within the loop

```
outloop EXPR
```

reloop based on the rounded value of the tag expression, causes branch back to the previous -loop- command at the same level of indentation without terminating the loop; rounded value < Ø causes branch to the previous -loop- at the same level; rounded value ≥ Ø causes fall-through to the following command within the loop

```
reloop  EXPR
```

NOTE: Commands between -loop- and -endloop- except -outloop- and -reloop- must be indented and marked by the indent symbol. The -outloop- and -reloop- commands must be at the same level of indentation as the -loop- and -endloop- which mark the range of the loop.

Following is an example demonstrating placement of these commands.

```
loop    n1<1Ø
.       write   within loop         $$ executed if n1<1Ø
.       subl    n1                  $$ executed if n1<1Ø
reloop  n1≥5                        $$ executed if n1<1Ø
.       write   still within loop   $$ executed if n1<5
.       do      someunit            $$ executed if n1<5
outloop n1<3                        $$ executed if n1<5
.       write   still within loop   $$ executed if 3≤n1<5
endloop
write   outside of loop             $$ executed if n1≥1Ø or n1<3
```

Key-initiated sequencing

NOTE: The following commands (-next- through -lablop-) may have the conditional form, where argument x leaves the pointer unchanged, and argument q clears the pointer and renders the key inactive (except for NEXT, which causes fall-through to the following unit). Argument q is not valid with -nextnow-. Up to 100 arguments are permitted in the conditional tag. The conditional expression is evaluated when the command is executed, not when the key is pressed.

next, nextl, back, backl, stop specifies the unit executed when the student presses the appropriate key (arrows must be satisfied before branch on the NEXT key)

```
next    UNIT NAME
backl   UNIT NAME
back    (B) or back    q   (clears back pointer; disables BACK key)
```

nextnow terminates processing in the unit and makes only NEXT key active

```
nextnow UNIT NAME
```

nextop, nextlop, backop, backlop specifies the unit executed when the student presses the appropriate key; there is no full-screen erase and new information is plotted on-the-page; the unit specified, however, is a main unit

```
nextlop UNIT NAME
backop  (B) or backop  q   (clears back pointer; disables BACK key)
```

help, helpl, data, datal, lab, labl initiates a help-type sequence by specifying the unit to be executed if the student presses the appropriate key; sets the base pointer for the unit to return to unless the base pointer is already set; the unit branched to is a main unit but not a base unit (unless the base pointer is reset to this unit); a help-type sequence may be terminated by the -end- command

```
help    UNIT NAME
lab     (B) or lab     q   (clears lab pointer; disables LAB key)
```

helpop, helplop, dataop, datalop, labop, lablop specifies the unit executed when the student presses the appropriate key; the unit branched to is not a main unit or a base unit and no full-screen erase is performed; control is returned to the main unit after execution of the helpop-type unit

```
helpop  UNIT NAME
dataop  (B) or  dataop  q   (clears data pointer; disables DATA key)
```

term permits use of the TERM key to initiate a help-type sequence starting at the unit containing this command and the specified character string; sequence can be terminated by -end- (see -inhibit term-)

```
term    STRING   (maximum of 8 characters)
term    (B)   (provides match to any term request that does not
              match an author-specified or system-specified term)
```

termop similar to -term- except initiates a helpop-type branch

```
termop  STRING   (maximum of 8 characters)
termop  (B)
```

NOTE: A lesson may have up to 299 -term- and -termop- commands.

base resets or clears the base pointer in order to alter help-type sequencing

```
base    (B) or base    q   (clears base pointer)
base    UNIT NAME          (sets base pointer to named unit)
base    EXPR,q,NAME,x      (example of conditional form;
          argument x leaves base pointer unchanged; argument q
          clears base pointer; maximum of 100 arguments in
          conditional tag)
```

end terminates a help-type sequence or a lesson

```
end     (B) or end     help  (ends a help-type sequence; may
          occur anywhere in a unit; student is returned to the
          base unit after pressing NEXT; -end- is ignored in a
          non-help-type sequence)
end     lesson  (when NEXT is pressed after execution of this
          statement, the student is returned to the router lesson
          or to the "Press NEXT to Begin" page; finish unit is not
          executed; authors are returned to the author-mode page)
```

Timing

group — (non-executable) forms a group of keys with the specified name for use with -pause- and -keytype- commands

```
group   NAME,KEY1,KEY2,KEY3,...   (from 2 to 7 characters in NAME)
group   NAME,NAME1,NAME2,...      (groups may be combined)
```

Note: System-defined groups are:

```
alpha     (letters:  a to z and A to Z)
numeric   (digits: Ø to 9)
funct     (function keys)
touch     (input from touch panel)
ext       (input from external device other than touch panel)
all       (any keypress, touch-panel input, or external input)
```

pause — delays execution of subsequent commands by the specified interval or until the specified keys are pressed

```
pause   EXPR GIVING NUM SECONDS   (minimum of .75 second)
pause   Ø   (causes no pause; exception to .75 second minimum)
pause   (B) or pause   NEGATIVE VALUE   (interrupts processing
           until any keypress comes in)
pause   keys=KEY1,KEY2,GROUP NAME,...   (interrupts processing
           until one of the specified keys comes in; all keynames
           are typed without quote marks and function keys are
           typed in lower case)
pause   NUM SECONDS,keys=KEY1,KEY2,GROUP NAME,...   (interrupts
           processing for the specified time or until one of the
           specified keys comes in)
```

Note: If a function key other than next, such as help, is specified and there is a preceding -help- or -helpop- command specifying a unit to execute, this unit is executed rather than the command following the -pause-. If next is specified, the NEXT key just breaks the -pause-, even if there is a preceding -next- command.
The statement -pause keys=touch- enables the touch panel.

collect — allows storage of keycodes from keyset, touch panel, or external inputs in successive variables, starting at the specified variable; collection terminates with receipt of the specified number of keys or with receipt of the TIMEUP key, which is also stored

```
collect STARTING VAR,NUM KEYS   (must use student variables)
```

keytype sets a variable according to the position in a list of the key pressed by the student; if the key pressed is not listed, the variable is set to -1

keytype VAR,ARGØ,ARG1,ARG2,...

arguments ARGØ, ARG1, ARG2,... may be any of the following:

KEYNAME	(any keyname; no quotation marks are used; function keys are in lower case)
GROUP NAME	(name of a system-defined group or of a group set up by the -group- command)
(VAR)	(value of "key" is compared with the value stored in VAR)
ext(VAR)	(when the 1Øth bit from the right of "key" equals 1, indicating an external input, the right-most 9 bits of "key" are compared with the value stored in VAR)

touch(COARSE;WIDTH IN CHARACTERS,HEIGHT IN LINES)
touch(FINEX,FINEY;WIDTH IN DOTS,HEIGHT IN DOTS)
(COARSE or FINEX,FINEY is the screen position of the lower left corner of a rectangle with specified width and height; width and height are optional and are assumed to be Ø if omitted)

Note: Up to 1ØØ keys may be specified; groups count as one key.

time presses the TIMEUP key after the specified interval and sets "key" to "timeup"; function keys can break through the timing and set "key" to the key pressed

time EXPR GIVING NUM SECONDS (minimum of .75 second)
time (B) or time NEGATIVE VALUE (clears any -time- in effect)

timer specifies a unit in the router to which a routed student is sent when the indicated time has elapsed

timer NUM SECONDS,UNIT NAME (minimum of 3ØØ seconds)
timer (B) (clears any -timer- in effect)

timel specifies a unit in the same lesson to execute (via helpop-type branch) when the indicated time has elapsed; remains in effect across other timing commands and across unit boundaries

timel NUM SECONDS,UNIT NAME (minimum of .75 second)
timel (B) (clears any -timel- in effect)

press puts the specified key into the student input buffer for the indicated station, if given; limited to one keypress per second

```
press   KEYCODE
press   "KEYNAME"   (for non-function keys)
press   KEYNAME     (for function keys, e.g., press   next)
press   KEYCODE,STATION   (presses the key at another station
           if that station is in the same lesson as the -preus-
           command; "zreturn" is set to -1 if the station is in
           the lesson, to Ø if not)
```

catchup (no tag) causes a pause in execution while transmission of accumulated output to the terminal is completed in order to synchronize display and execution of commands

return (no tag) interrupts processing and returns with a new time slice for further processing

cpulim specifies the maximum CPU usage rate in thousand instructions per second with a maximum of 1Ø thousand instructions per second

```
cpulim  EXPR GIVING MAXIMUM CPU USAGE RATE   (maximum of 1Ø)
```

Lesson connections and sections

use (non-executable) inserts the specified block(s) from the lesson specified on the lesson information page into the lesson being condensed; all contiguous blocks with the same name are taken; use codes on the lessons must match

use BLOCK NAME

jumpout causes execution of commands in the specified lesson, in a specific unit in that lesson, if given, otherwise in the first unit (see -inhibit jumpchk- and -inhibit from-)

jumpout LESSON NAME (goes to the first unit in the lesson; jumpout codes need not match; variable tag must be enclosed in parentheses)
jumpout LESSON NAME,UNIT NAME (jumpout codes must match)
jumpout return (returns to the first unit of the lesson from which a jumpout was made to the present lesson)
jumpout return,return (returns to the lesson from which a jumpout was made to the unit following the unit containing the -jumpout- command)
jumpout (B) or jumpout q (causes a jumpout to the author-mode page for authors and to "Press NEXT to begin" page or to a router for students; similar to -end lesson-)
jumpout <LESLIST POSITION> (causes a jumpout to the first unit of the lesson at the specified position in the leslist)
jumpout <LESLIST POSITION>,UNIT NAME (causes a jumpout to the specified unit in the lesson at the specified leslist position; variable unit name must be enclosed in parentheses; jumpout codes must match)
jumpout EXPR;LESSONM,UNITM;LESSONØ;LESSON1,UNIT1;q;x (example of conditional form; argument q causes jumpout as above; argument x causes no jumpout)
jumpout resume (allows router lesson to return the student to the lesson and unit specified by the last -restart-)
jumpout continue (used in a site lesson to send the user to his router or lesson)
jumpout GNOTEFILE NAME (causes jumpout to the specified group notes file; return to lesson is automatic, via -jumpout return,return-)
jumpout notes,choice (causes jumpout to index page of student notes with read and write access)
jumpout notes,read (allows read access to student notes)
jumpout notes,write (allows write access to student notes)
jumpout notes,instruct (allows user to read student notes as an instructor)

from checks the lesson and main unit, if specified, from which a lesson was entered against a list, and sets a variable to the relative position of the lesson and unit in the list; if the lesson and unit are not listed, the variable is set to -1 (if no unit is specified, any unit in the lesson qualifies); alternate form stores lesson name and unit name in specified variables

for example:

```
from    VAR;LESSONØ,UNITØ;LESSON1;<LESLIST POSITION>,UNIT2;
from    VAR FOR LESSON NAME,VAR FOR UNIT NAME (opt)
```

lessin checks if the lesson specified is credited to the user's logical site; sets "zreturn" to -1 if the lesson is in ECS and in use at the user's logical site and to Ø otherwise

```
lessin  'LESSON NAME'
lessin  (VAR CONTAINING LESSON NAME)
lessin  <LESLIST POSITION OF LESSON>
```

in sets "zreturn" to indicate whether the specified station number is in the lesson containing the -in- command

```
in      EXPR   (value from Ø to 1151)

zreturn = -2  if the -in- command is in a router lesson and a
              routed student at the specified station is in an
              instructional lesson
        = -1  if the station is in the instructional lesson
              containing the -in- command
        =  Ø  if the station is not in the lesson
```

notes initiates TERM-comments automatically, or sends specified text to the lesson notes file or student notes file without user interaction

```
notes   (B)
notes   STARTING VAR CONTAINING TEXT,NUM VARS   (inserts text at
           front of note)
notes   STARTING VAR CONTAINING TEXT,NUM VARS,send   (sends the
           text without user interaction)

Note:  Student variables must be used for the text; the format
       is that for -text- command.

       zreturn = -1  if the note was sent successfully
               =  Ø  if the user pressed BACK1, and note was not sent
               =  1  if TERM-comments is not allowed in the lesson
               =  2  if the format of the text is incorrect
               =  3  if there is an error in connecting the notes file
```

cstart (non-executable) (no tag) indicates subsequent code is to be condensed (used after a preceding -cstop-)

cstop (non-executable) (no tag) indicates subsequent code is not to be condensed; in effect up to the next -cstart-, if any

cstop* (non-executable) (no tag) indicates none of the subsequent code is to be condensed, independent of subsequent -cstart- commands

NOTE: It is preferable to use the partial condense option of the editor rather than -cstart-, -cstop-, and -cstop*-.

Lesson lists

leslist references two or three special blocks containing a list of up to 320 or 480 lessons (numbered 0 through 319 or 0 through 479) available for general use; common codes on the lesson containing the leslist blocks and the lesson containing the -leslist- command must match when leslist blocks and -leslist- command are in different lessons

```
leslist (Ø),LESLIST NAME
leslist (zlesson),LESLIST NAME
leslist ,LESLIST NAME
leslist LESLIST NAME
leslist LESSON NAME,LESLIST NAME  (LESSON NAME contains the leslist
          blocks; enclose variable arguments in parentheses)
```

Note: zreturn = -1 if the -leslist- command is executed successfully
= 0 if the leslist blocks are not found
= 1 if the common codes do not match
= 2 if this is a duplicate leslist reference

addlst allows addition of a lesson name to a leslist, either in the specified slot or in the first empty slot if none is specified; the leslist is not edited directly, i.e., the name is added from student mode; the tag must be a variable; requires three consecutive variables (the name is stored with: -storea STARTING VAR;3Ø-)

```
addlst  STARTING VAR,LESLIST POSITION (opt)
```

Note: zreturn = -1 if the lesson name is added successfully
= 0 if there is no preceding successful -leslist- command
= 1 if the form of the lesson name is incorrect
= 2 if the lesson name is already in the leslist (with one-argument form only)
= 3 if the leslist is full
= 4 if the specified slot is occupied (with two-argument form only)

removl allows deletion of a lesson at a specified leslist position from student mode (similar to -addlst-); the vacated position is left blank

```
removl  LESLIST POSITION
```

Note: zreturn = -1 if the lesson is removed successfully
= 0 if there is no preceding successful -leslist- command

lname stores the lesson name at the specified leslist position in three consecutive variables starting at the specified variable

```
lname   STARTING VAR,LESLIST POSITION
```

Note: Use with -showa- to display the lesson name; e.g.:

```
lname   STARTING VAR,LESLIST POSITION
showa   STARTING VAR,3Ø
```

zreturn = -1 if execution is successful
= Ø if there is no preceding sucessful -leslist- command

findl searches the leslist for the lesson name stored in three consecutive variables and returns the leslist position in the specified variable

```
findl   STARTING VAR FOR LESSON NAME,VAR FOR LESLIST POSITION
```

Note: If the lesson name is not found or if no leslist is used, returned value for the leslist position is -1.

Lesson annotation and debugging

*
c(space) indicates the statement on that line is a comment only and is to be ignored by the computer

```
*This is a comment.
c This is a comment.
```

$$ (not a command) when placed on the same line with a TUTOR statement indicates that subsequent material on that line is a comment

```
COMMAND TAG    $$this is a comment
```

change (non-executable) permits names of commands to be changed, e.g., to a language other than English; also permits symbols (e.g., punctuation) to be redefined in certain judging commands; -change- must be placed in the initial entry unit; all changes are in effect for the entire lesson and cannot be altered

```
change  command NORMAL TUTOR NAME to NEW NAME
change  symbol SYMBOL1 to SYMBOL2
```

for example:

```
change  command at to wo
change  symbol * to letter
change  symbol ? to puncword
change  symbol p to punc
change  symbol 3 to vowel
change  symbol a to b
change  symbol space to letter
change  symbol sup to null
    etc.
```

Note: The answer-matching commands affected by -change symbol- are: -answer-, -wrong-, -answerc-, -wrongc-, -concept-, -miscon-, -match-, -storen-. Other commands affected are: -getword-, -getmark-, -getloc-, and -compare-.

step allows an author to step through a lesson command by command; when accessed through TERM-step, the author's security code must match the lesson's change code (not available with student records)

```
step    on    or   step    EXPR    (EXPR≠Ø turns on step option)
step    off   or   step    EXPR    (EXPR=Ø turns off step option)
```

*list (does not affect condensing or execution) allows certain print options for listing a lesson; -*list- commands for printing special types of blocks must precede those blocks in the program

*list charset (prints contents of the charset in the lesson with 0 for dots on and - for dots off)
*list charset,(DOTSBLANKS) (prints contents of charset with symbols specified for dots and blanks)
*list commands,COMMAND1,COMMAND2,COMMAND3,... (up to 10 commands; lists lines on print where specified commands appear)
*list common,COMMON NAME,NUM WORDS,FORMAT (prints contents of common; see Note on next page for information on format)
*list deleted (prints deleted lines [with "mod words" option])
*list eject (causes page eject where command is located)
*list ignore (causes all subsequent -*list- commands to be ignored)
*list info (prints lesson information display)
*list label,YOUR LABEL INCLUDING SPACES (prints a label at the location of the command)
*list leslist (prints the contents of all leslists in the lesson)
*list micro (prints the contents of all microtables in the lesson)
*list mods (prints "mod words": first 5 characters of the name of the last person to change each line and the date of the change; available when the mod words option is turned on for the lesson)
*list off (stops printing source code at the location of the command and starts printing unit cross-reference table and any preceding -*list- options)
*list off,BLOCKNAME1,BLOCKNAME2,BLOCKNAME3-BLOCKNAME4 (specifies blocks that are not to be printed)
*list parts (prints only those blocks which have been set to condense)
*list symbols (prints reference table of variables, defined and primitive, used in the lesson)
*list text (prints only text of -write- and -writec- commands)
*list title,YOUR TITLE (specifies subheading to be printed under the lesson name on each page)
*list vocab (prints contents of vocab blocks)

Instructions for printing datasets are specified on the data page of the dataset. Directives include: page eject instructions (with special format), if any, starting record number, number of records, and format. The formats are the same as formats for printing common.

See next page for formats for printing common and datasets.

Note: For printing datasets and commons, FORMAT may be:

integer	or i	(nc-variables; prints 10 words per line)
exponential	or e	(vc-variables; prints 10 words per line)
octal	or o	(prints 5 words per line)
alpha	or a	(prints 10 words per line)
x		(prints each word in all formats; prints 2 words per line)
special	or s	(special format; specified number of words to be printed is ignored but field must be present; see details below)
(DESIGNED)		(designed format; enclosed in parentheses; see details below)

special format:

words are interpreted in alpha;
words with all 0 bits are ignored unless they are preceded by at least one non-zero word on the same line;
a word ending with at least 12 zero bits signals the end of a line;
up to 120 characters may be printed per line;

the following print options are placed directly in the dataset or common; each requires two consecutive words:

*format blocks	(signals a page eject after each subsequent block)
*format eject	(causes a page eject at the location of this directive)
*format end	(indicates the print is to end at the location of this directive)
*format pages	(signals page eject after each printed page; allows top and bottom margins)
*format records	(signals a page eject after each subsequent dataset record)

designed format:

the format is for a line of print;
format must be enclosed in parentheses;
up to 135 characters may be printed per line;

formats available:

i (integer; nc-variables; prints 10 characters per word)
e (exponential; vc-variables; prints 10 characters per word)
o (octal; prints 20 characters per word)
a (alpha; prints 10 characters per word)
x (space; preceding number indicates number of spaces)
l (common or dataset location; prints 4 characters)
p (skip to next word to be printed on the same line; preceding number indicates how many words to go forward)
commas and spaces may be added for readability

System variables for sequencing

args — number of arguments transferred at the previous execution of a unit with arguments

backout
= -2 for a single-station backout
= -1 for a general backout
= Ø for no backout (e.g. signoff via STOP1)
= +1 after -station stop1-

baseu — name of the user's current base unit or
= Ø if no base unit is specified, indicating the user is not in a help-type sequence

clock — value of the system clock in seconds (to the nearest millisecond) since the previous deadstart (see command -clock-)

fromnum — leslist position of the lesson from which the user came via a jumpout; = -1 if the lesson is not in a leslist or if no leslist is being used

key — after a keyset input: contains the 7-bit keycode of the last keypress; after a touch-panel input: contains a 9-bit number which gives the location of the touch square (the binary form of this number is 1xxxxyyyy, where the 4 bits labeled "x" give the horizontal touch location and the 4 bits labeled "y" give the vertical touch location--coordinates for touch squares on the screen are: Ø,Ø at lower left, Ø,15 at upper left, 15,Ø at lower right, 15,15 at upper right); after an external input: contains a 1Ø-bit number whose left-most 2 bits are 1Ø, with the remaining 8 bits carrying information from the external source

lessnum — leslist position of the user's current lesson; = -1 if the lesson is not in the leslist or if no leslist is being used

lleslst — number of lessons the leslist will hold (=Ø if no leslist is in use)

llesson — condensed length of the lesson

mainu — name of the user's current main unit

mallot — memory allotment for the logical site at which the user is working

muse — total memory usage by users at the same logical site as the user

nhelpop number of times a help-type key is pressed for on-the-page help; zeroed for each main unit and for each arrow in the unit

proctim CPU usage in seconds (accurate to 1 millisecond)

ptime = -1 if the current time is during prime-time hours
= Ø otherwise

sitenam name of the user's logical site

station identification number assigned by the system to a terminal
physical site = station ars 5
site station # = station $mask$ o37
station = 32 × physical site + site station #

tactive number of currently active terminals

user user type: author, instructor, student, multiple, sabort (if student records have been aborted), snockpt (if automatic checkpoint has been aborted)

usersin number of users in the lesson (routed students are counted as being in the router as well as in the instructional lesson)

zaccnam name of the account which contains the user's course

zcondok = -1 if the lesson condenses without errors or warnings
= Ø if the lesson has condense errors or warning messages

zfroml name of the lesson from which a jumpout was done

zfromu name of the unit from which a jumpout was done

zlesson name of user's current lesson

zpnfile = -1 if the user's course has a personal notes file attached
= Ø otherwise (and for students and multiples)

zpnotes = -1 if the user has new, unread personal notes
= Ø otherwise

zretrnu name of the unit to which -jumpout return,return- will go

zreturn general-purpose system variable--set with certain commands according to the results of an operation (see -addlst-, -addname-, -addrecs-, -charset-, -chartst-, -commonx-, -comret-, -datain-, -dataout-, -dataset-, -delrecs-, -ext-, -in-, -leslist-, -lessin-, -lname-, -micro-, -names-, -notes-, -press-, -readd-, -readr-, -readset-, -release-, -removl-, -rename-, -reserve-, -setname-, -site-, station-)

zsnfile = -1 if student user's course has a student notes file attached
= Ø otherwise (and for authors and instructors)
= +1 if access to student notes and lesson notes is not allowed

zsnotes = -1 if the student has new, unread notes
= Ø otherwise

zterm contains last term requested by the user

ztouchx fine-grid x-location of the center of the touch box touched (= -1 if last input was not a touch input)

ztouchy fine-grid y-location of the center of the touch box touched (= -1 if last input was not a touch input)

zunit name of the user's current unit

zwpb number of computer words per block in a lesson (currently 32Ø)

NOTE: The following system variables contain alphabetic information (left-justified) and must be displayed with -showa-:
"baseu", "mainu", "sitenam", "user", "zaccnam", "zfroml", "zfromu", "zlesson", "zretrnu", "zterm", and "zunit".

In addition to the system variables listed in this subsection, keynames of function keys may be treated as system constants. These keynames are typed in lower case (e.g., next, lab, term) and have the numerical values given in the keycode table in the appendix. The exception is the SQUARE key, which has the keyname "microl".

Some limits associated with commands

area	1Ø characters in name of area
argume[illegible] unit	1Ø arguments
arheada	five 6-bit characters
bump	8 characters
calcc calcs	61 calculations
common	8ØØØ variables (>15ØØ variables requires -comload-)
comload	15ØØ variables
compute	1ØØ characters in character string
conditional form of commands which reference unit names, e.g., -next-, -help-, etc.	1ØØ unit names (line containing 1Ø1st name is flagged as a condense error)
course	8 characters in sign-on course name
cpulim	value of tag from 1 to 1Ø
define	7 characters in name of define set 7 characters in name of variable 4ØØ definitions (or less if definitions are long) 6 arguments in defined function 1Ø units (e.g., kg, m, sec, liter, etc.) 255 elements in an array
delay	1 second maximum
deletes	value of increment from 1 to 5ØØ
do join } combined	1Ø levels
draw gdraw rdraw	63 arguments
edit	maximum of 3ØØ characters in buffer
endings	1Ø separate -endings- commands, 8 endings in each tag
finds findsa	value of increment from 1 to 5ØØ

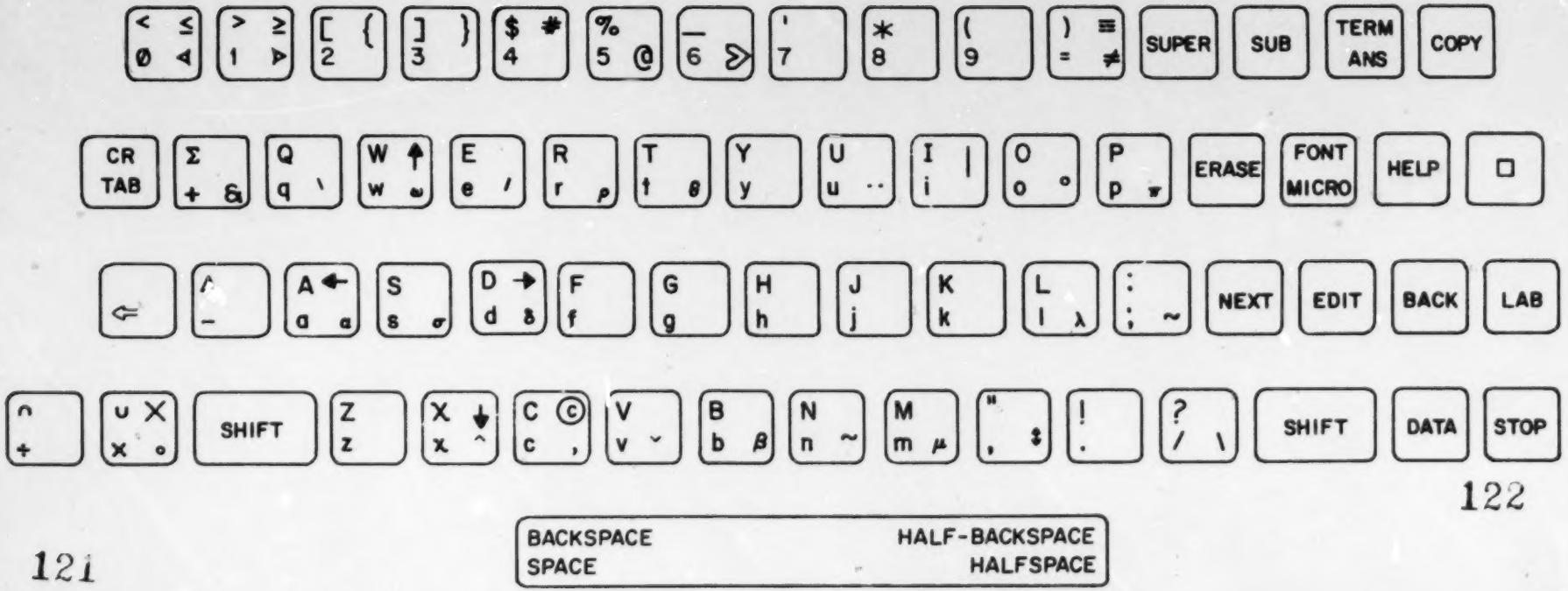
graph	9 characters in string to be plotted
group	from 2 to 7 characters in the name
hbar vbar	9 characters in string to be plotted
in	value of tag from Ø to 1151
inserts	increment between entries × number of entries to add ≤ 5ØØ
keytype	1ØØ keys in tag (groups count as one key)
list	7 characters in name of list
long	3ØØ characters (>15Ø requires -edit- for active EDIT key)
move	5ØØØ characters may be moved
name	18 characters in sign-on name
noword okword	9 characters in word
outputl	1Ø characters in label, 2Ø consecutive variables
pack	5ØØ characters in character string
packc	1ØØ character strings character count is limited by length of line of code
pause	.75 second minimum
press	1 keypress per second
put putd putv	5Ø characters in strings; expansion due to string replacement limited to 3ØØ characters
routvar	5Ø variables
score	value of tag from Ø to 1ØØ
set	up to 61 separate values
setdat	value for "atime" less than elapsed time for the session values for "aarrows", "ahelp", etc. less than 512
setperm	(one-argument) integer from Ø to 12Ø (two-argument) integer from Ø to 3ØØØ
sort sorta	value of increment from 1 to 2ØØ

storage		1500 variables
term termop		8 characters in string, 299 terms in a lesson
time		.75 second minimum
timel		.75 second minimum
timer		300 seconds minimum
transfr		length is the smaller of: size of common or storage (reference to ECS) or length of -comload- or -stoload- (reference to CM) or 150 (reference to student variables) or tag of -routvar- (reference to router variables)
unit entry	combined	8 characters in name 500 condensed words in a unit 395 distinct condensed units in a lesson
vocab vocabs		7 characters in name of vocabulary
*list commands		10 commands

Standard PLATO Keyset

GUIDE

Upper Case	Access, Upper Case
Lower Case	Access, Lower Case



KEYCODES

(the keycode is placed in the system variable "key" when the key is pressed)

key	keycode	key	keycode	key	keycode
a	1 = o01	,	46 = o56	_	97 = o141
b	2 = o02	.	47 = o57	'	98 = o142
c	3 = o03	+	48 = o60	Σ	101 = o145
d	4 = o04	[	49 = o61	Δ	102 = o146
e	5 = o05	]	50 = o62	?	104 = o150
f	6 = o06	%	51 = o63	"	110 = o156
g	7 = o07	×	52 = o64	!	111 = o157
h	8 = o10	÷	53 = o65	∩	112 = o160
i	9 = o11	sub	54 = o66	∪	116 = o164
j	10 = o12	super	55 = o67	locksub	118 = o166
k	11 = o13	shift÷	56 = o70	locksup	119 = o167
l	12 = o14	car ret	57 = o71	:	127 = o177
m	13 = o15	<	58 = o72	FUNKEY	128 = o200
n	14 = o16	>	59 = o73	NEXT	130 = o202
o	15 = o17	bkspace	60 = o74	NEXT1	131 = o203
p	16 = o20	font	61 = o75	ERASE	132 = o204
q	17 = o21	access	62 = o76	ERASE1	133 = o205
r	18 = o22	;	63 = o77	HELP	134 = o206
s	19 = o23	A	65 = o101	HELP1	135 = o207
t	20 = o24	B	66 = o102	BACK	136 = o210
u	21 = o25	C	67 = o103	BACK1	137 = o211
v	22 = o26	D	68 = o104	LAB	138 = o212
w	23 = o27	E	69 = o105	LAB1	139 = o213
x	24 = o30	F	70 = o106	DATA	140 = o214
y	25 = o31	G	71 = o107	DATA1	141 = o215
z	26 = o32	H	72 = o110	TERM	142 = o216
0	27 = o33	I	73 = o111	ANS	143 = o217
1	28 = o34	J	74 = o112	COPY	144 = o220
2	29 = o35	K	75 = o113	COPY1	145 = o221
3	30 = o36	L	76 = o114	EDIT	146 = o222
4	31 = o37	M	77 = o115	EDIT1	147 = o223
5	32 = o40	N	78 = o116	MICRO	148 = o224
6	33 = o41	O	79 = o117	SQUARE	149 = o225
7	34 = o42	P	80 = o120	STOP	150 = o226
8	35 = o43	Q	81 = o121	STOP1	151 = o227
9	36 = o44	R	82 = o122	TAB	152 = o230
+	37 = o45	S	83 = o123	TIMEUP	155 = o233
-	38 = o46	T	84 = o124	CATCHUP	157 = o235
*	39 = o47	U	85 = o125	touch-	256 = o400
/	40 = o50	V	86 = o126	panel	↓ ↓
(	41 = o51	W	87 = o127	inputs	511 = o777
)	42 = o52	X	88 = o130	external	512 = o1000
$	43 = o53	Y	89 = o131	inputs	↓ ↓
=	44 = o54	Z	90 = o132		767 = o1377
space	45 = o55				

INTERNAL CODES

unshifted char	code	shifted char	code	access char	code	access-shifted char	code
a	o01	A	o7001	α	o7601	←	o767001
b	o02	B	o7002	β	o7602		
c	o03	C	o7003	´	o7603	©	o767003
d	o04	D	o7004	δ	o7604	→	o767004
e	o05	E	o7005	˙	o7605		
f	o06	F	o7006				
g	o07	G	o7007				
h	o10	H	o7010				
i	o11	I	o7011			↓	o767011
j	o12	J	o7012				
k	o13	K	o7013				
l	o14	L	o7014	λ	o7614		
m	o15	M	o7015	μ	o7615		
n	o16	N	o7016	˜	o7616		
o	o17	O	o7017	°	o7617		
p	o20	P	o7020	π	o7620		
q	o21	Q	o7021	ˋ	o7621		
r	o22	R	o7022	ρ	o7622		
s	o23	S	o7023	σ	o7623		
t	o24	T	o7024	θ	o7624		
u	o25	U	o7025	¯	o7625		
v	o26	V	o7026	¨	o7626		
w	o27	W	o7027	ω	o7627	↑	o767027
x	o30	X	o7030	ˆ	o7630	↓	o767030
y	o31	Y	o7031				
z	o32	Z	o7032				
0	o33	left64$^{\$}$	o7033	⟨	o7633		
1	o34	left$^{\$}$	o7034	⟩	o7634		
2	o35						
3	o36						
4	o37						
5	o40	right$^{\$}$	o7040	⊕	o7640		
6	o41	_	o7041	⩾	o7641		
7	o42	'	o7042				
8	o43						
9	o44						
+	o45	Σ	o7045	&	o7645		
-	o46	Δ	o7046				
*	o47						
/	o50	?	o7050	\	o7650		
(	o51						
)	o52			≡	o7652		
$	o53			≠	o7653		
=	o54			≠	o7654		
space	o55			.5space	o7655		

INTERNAL CODES (cont.)

unshifted char	code	shifted char	code	access char	code	access-shifted char	code
,	o56	"	o7056	♦	o7656		
.	o57	!	o7057				
+	o60	¤	o7060				
[	o61			{	o7661		
]	o62			}	o7662		
%	o63						
×	o64	∪	o7064	∘	o7664	×	o767064
♦	o65			cr 01*	o7665		
sub	o66	locksub	o7066	dnlin*	o7666	ldnlin*	o767066
super	o67	locksup	o7067	uplin*	o7667	luplin*	o767067
shift	o70						
car ret*	o71	ldnlin*	o7071	cr101*	o7671		
<	o72			≤	o7672		
>	o73			≥	o7673		
bkspace	o74			.5bksp	o7674		
font	o75						
access	o76						
;	o77	:	o7077	~	o7677		

*NOTE:
left64: write to left, starting at character position 64
left: write to left, starting at "where" position
right: write to right, starting at "where" position
cr 01: carriage return to next line, character position 1
dnlin: write next character down one line (down 16 dots)
ldnlin: same as dnlin except locking
uplin: write next character up one line (up 16 dots)
luplin: same as uplin except locking
car ret: standard carriage return to next line, left margin
cr101: carriage return to screen position 101

The following characters, although produced with the shift key, do not produce a shift code in the internal code: *, (,), $, [,], %, <, >, car ret, backspace, and font. In this table these characters are in the "unshifted" category.

NOTE: Information on numerical keycodes and internal codes may be provisional. Use these values for testing purposes only. Wherever possible, use key names (with or without quotes, as required).

ALTERNATE FONT TERMINAL MEMORY LOCATIONS

(key associated with terminal memory location)

loc	key	loc	key	loc	key	loc	key	loc	key
0	space	27	0	54	(k)	81	Q	108	)
1	a	28	1	55	(l)	82	R	109	(E)
2	b	29	2	56	(m)	83	S	110	(F)
3	c	30	3	57	(n)	84	T	111	(G)
4	d	31	4	58	(o)	85	U	112	∩
5	e	32	5	59	;	86	V	113	(H)
6	f	33	6	60	"	87	W	114	(I)
7	g	34	7	61	,	88	X	115	(J)
8	h	35	8	62	.	89	Y	116	∪
9	i	36	9	63	unav	90	Z	117	(/)
10	j	37	+	64	space	91	<	118	(K)
11	k	38	-	65	A	92	>	119	(L)
12	l	39	(a)	66	B	93	[	120	(M)
13	m	40	/	67	C	94	]	121	(N)
14	n	41	(b)	68	D	95	$	122	(O)
15	o	42	(c)	69	E	96	%	123	:
16	p	43	(d)	70	F	97	_	124	!
17	q	44	=	71	G	98	'	125	(!)
18	r	45	(e)	72	H	99	*	126	(.)
19	s	46	(f)	73	I	100	(	127	unav
20	t	47	(g)	74	J	101	Σ		
21	u	48	÷	75	K	102	Δ		
22	v	49	(h)	76	L	103	(A)		
23	w	50	(i)	77	M	104	?		
24	x	51	(j)	78	N	105	(B)		
25	y	52	×	79	O	106	(C)		
26	z	53	◆	80	P	107	(D)		

NOTE: unav: Location is unavailable on standard terminals.
(key): Press MICRO (or shift-square) followed by the indicated key.

Powers of 2

n	2^n		n	2^n	
ø	1	$=8^{ø}$	3ø	1 ø73 741 824	$=8^{1ø}$
1	2		31	2 147 483 648	
2	4		32	4 294 967 296	
3	8	$=8^{1}$	33	8 589 934 592	$=8^{11}$
4	16		34	17 179 869 184	
5	32		35	34 359 738 368	
6	64	$=8^{2}$	36	68 719 476 736	$=8^{12}$
7	128		37	137 438 953 472	
8	256		38	274 877 9ø6 944	
9	512	$=8^{3}$	39	549 755 813 888	$=8^{13}$
1ø	1 ø24		4ø	1 ø99 511 627 776	
11	2 ø48		41	2 199 ø23 255 552	
12	4 ø96	$=8^{4}$	42	4 398 ø46 511 1ø4	$=8^{14}$
13	8 192		43	8 796 ø93 ø22 2ø8	
14	16 384		44	17 592 186 ø44 416	
15	32 768	$=8^{5}$	45	35 184 372 ø88 832	$=8^{15}$
16	65 536		46	7ø 368 744 177 664	
17	131 ø72		47	14ø 737 488 355 328	
18	262 144	$=8^{6}$	48	281 474 976 71ø 656	$=8^{16}$
19	524 288		49	562 949 953 421 312	
2ø	1 ø48 576		5ø	1 125 899 9ø6 842 624	
21	2 ø97 152	$=8^{7}$	51	2 251 799 813 685 248	$=8^{17}$
22	4 194 3ø4		52	4 5ø3 599 627 37ø 496	
23	8 388 6ø8		53	9 øø7 199 254 74ø 992	
24	16 777 216	$=8^{8}$	54	18 ø14 398 5ø9 481 984	$=8^{18}$
25	33 554 432		55	36 ø28 797 ø18 963 968	
26	67 1ø8 864		56	72 ø57 594 ø37 927 936	
27	134 217 728	$=8^{9}$	57	144 115 188 ø75 855 872	$=8^{19}$
28	268 435 456		58	288 23ø 376 151 711 744	
29	536 87ø 912		59	576 46ø 752 3ø3 423 488	

Given the byte size = n: range for unsigned integers is ø to 2^n-1

range for signed integers is $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

Given the maximum absolute value such that $2^{n-1} \leq |\text{maximum}| < 2^n$:

byte size for unsigned integers is n

byte size for signed integers is n+1

Alphabetical index to system variables

These system variables have been classified (in some cases, approximately) according to their functions. They may be used wherever expressions are accepted, e.g., in tag of -calc-, -at-, etc.

Word	Page	Word	Page	Word	Page
aarea	D7	lstatus	R3	zaccnam	S21
aarrows	D7	lstorage	C24	zbpc	C24
ahelp	D7	mainu	S2Ø	zbpw	C24
ahelpn	D7	mallot	S2Ø	zcondok	S2Ø
anscnt	J19	mode	P2Ø	zcpw	C24
ansok	J19	muse	S2Ø	zcusers	C24
aok	D7	nhelpop	S21	zdsname	C24
aokist	D7	ntries	J19	zdusers	C24
args	S2Ø	opcnt	J21	zfroml	S21
asno	D7	order	J2Ø	zfromu	S21
aterm	D7	phrase	J2Ø	zinfo	C24
atermn	D7	proctim	S21	zlesson	S21
atime	D8	ptime	S21	znscpn	C24
auno	D8	rcallow	R3	znsmaxn	C24
backout	S2Ø	router	R3	znsmaxr	C24
baseu	S2Ø	rstartl	R3	znsnams	C24
capital	J19	rstartu	R3	znsrecs	C24
clock	S2Ø	rvallow	R3	zpnfile	S21
dataon	D7	sitenam	S21	zpnotes	S21
entire	J19	size	P2Ø	zrecs	C24
errtype	R3	sizex	P2Ø	zretrnu	S22
extra	J2Ø	sizey	P2Ø	zreturn	S22
formok	J2Ø	spell	J2Ø	zsessda	D7
fromnum	S2Ø	station	S21	zsesset	D7
jcount	J19	tactive	S21	zsesspt	D7
judged	J19	user	S21	zsnfile	S22
key	S2Ø	usersin	S21	zsnotes	S22
lcommon	C24	varcnt	J21	zterm	S22
ldone	R3	wcount	J2Ø	ztouchx	S22
lessnum	S2Ø	where	P2Ø	ztouchy	S22
lleslst	S2Ø	wherex	P2Ø	zunit	S22
llesson	S2Ø	wherey	P2Ø	zwpb	S22
lscore	R3			zwpr	C24

C D J M P R S A I

Alphabetical index to commands and related directives

Command	Page	Command	Page	Command	Page	Command	Page
abort	C17	commonx	C18	ext	P16	jumpout	S12
addlst	S15	compare	J14	extout	P16	keytype	S1Ø
addl	C3	comret	C17	find	C15	lab	S7
addname	C21	compute	C9	findall	C15	labl	S7
addrecs	C22	concept	J1Ø	findl	S16	labelx	P11
allow	R1	copy	J3	finds	C13	labely	P11
altfont	P19	course	C9	findsa	C13	labop	S8
ans	J12	cpulim	S11	finish	D6	lablop	S8
ansu	J11	cstart	S14	force	J1	leslist	S15
ansv	J11	cstop	S14	foregnd	C23	lessin	S13
answer	J9	cstop*	S14	from	S13	lesson	R2
answerc	J1Ø	data	S7	funct	P15	lineset	P19
array	C1	datal	S7	gat	P12	list	J4
area	D2	datain	C19	gatnm	P12	lname	S16
argument	S1, S2	dataoff	D1	gbox	P13	loada	J5
arheada	J2	dataon	D1	gcircle	P13	long	J2
arrow	J2	dataop	S8	gdot	P13	loop	S6
arrowa	J2	datalop	S8	gdraw	P13	lscalex	P11
at	P1	dataout	C19	getloc	J14	lscaley	P11
atnm	P1	dataset	C19	getmark	J13	markup	J18
audio	P16	date	C9	getname	C21	markx	P12
axes	P1Ø	day	C9	getword	J13	marky	P12
back	S7	define	C1	gorigin	P1Ø	match	J9
backl	S7	delay	P5	goto	S2	micro	P18
backgnd	C23	deletes	C14	graph	P13	miscon	J1Ø
backop	S7	delname	C23	group	S9	mode	P3
backlop	S7	delrecs	C22	gvector	P14	modperm	C8
base	S8	delta	P14	hbar	P14	move	C11
block	C16	disable	P16	help	S7	name	C9
bounds	P1Ø	do	S2	helpl	S7	names	C23
box	P6	dot	P6	helpop	S8	next	S7
branch	S4	doto	S4	helplop	S8	nextl	S7
bump	J5	draw	P6	hidden	P3	nextnow	S7
c	S16	edit	J2	htoa	C11	nextop	S7
calc	C2	else	S5	iarrow	J16	nextlop	S7
calcc	C2	elseif	S5	iarrowa	J16	no	J15
calcs	C3	embed	P1	ieu	S1	notes	S13
catchup	S11	enable	P16	if	S5	noword	J18
change	S17	end	S8	iferror	S3	ok	J15
char	P5	endarrow	J3	ignore	J15	okword	J18
charset	P18	endif	S5	imain	S3	open	J8
chartst	P19	endings	J4	in	S13	or	J12
circle	P7	endloop	S6	inhibit	P4	otoa	C11
circleb	P7	entry	S1	initial	C18	outloop	S6
clock	C9	erase	P3	inserts	C14	output	D2
close	J5	eraseu	J1	itoa	C11	outputl	D2
collect	S9	exact	J1Ø	jkey	J3	pack	C1Ø
color	P4	exactc	J1Ø	join	J16, S3	packc	C11
comload	C17	exactv	J11	judge	J17	pause	S9
commor	C17	exit	S3	jump	S2	play	P16

Alphabetical index to commands and related directives (cont.)

Command	Page	Command	Page	Command	Page	Command	Page
plot	P5	site	M1				
polar	P12	size	P4				
press	S11	slide	P16				
put	J5	sort	C12				
putd	J5	sorta	C12				
putv	J5	specs	J6				
randp	C7	station	M3				
randu	C7	status	R2				
rat	P8	step	S17				
ratnm	P8	stoload	C18				
rbox	P8	stop	S7				
rcircle	P9	storage	C18				
rdot	P8	store	J8				
rdraw	P8	storea	J8				
readd	D3	storen	J8				
readr	D5	storeu	J8				
readset	D3	subl	C3				
record	P16	tabset	P18				
release	C2Ø	term	S8				
reloop	S6	termop	S8				
remove	C7	text	P3				
removl	S15	time	S1Ø				
rename	C22	timel	S1Ø				
reserve	C2Ø	timer	S1Ø				
restart	D6	touch	J12				
return	S11	touchw	J12				
rorigin	P8	transfr	C16				
rotate	P4	unit	S1				
route	R1	use	S12				
routvar	R1	vbar	P14				
rvector	P9	vector	P6				
say	P17	vocab	J4				
sayc	P17	vocabs	J4				
saylang	P17	window	P7				
scalex	P1Ø	write	P1				
scaley	P11	writec	P1				
score	R2	wrong	J9				
search	C1Ø	wrongc	J1Ø				
seed	C7	wrongu	J11				
segment	C1	wrongv	J11				
set	C3	zero	C3				
setdat	D2						
setname	C21	*	S17				
setperm	C7	$$	S17				
show	P2	*format	S19				
showa	P3	*list	S18				
showe	P2						
showo	P3						
showt	P2						
showz	P2						